Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice

Agenda

AMD
Smarter Choice

9:00	Opening remarks	12:45	Tools (Part 2)
9:15	AMD Architecture	13:30	Compilers and Libraries
10:00	Coffee Break	14:30	Coffee Break
10:15	Multi-Threading	14:45	AMD CodeAnalyst™ demo
11:15	Tools (Part 1)	15:15	Q & A
12:00	Lunch and Celoxica demonstration	16:00	Closing remarks



AMD
Smarter Choice

Welcome

Simon Cole
Strategic Alliances Manager

Did you know…

Companies in the Fortune 500 develop on AMD

– Microsoft, Oracle, Verisign, and many more

AMD Core Math Library is downloaded ~1500 times each month

AMD Developer Central gets an average of 20,000 unique visitors a month from over 150 countries

AMD Opteron™ processor leads in key TPC and SPEC performance benchmarks

And to continue our support of developers,

introducing…

AMD Developer Days

What we are bringing you today

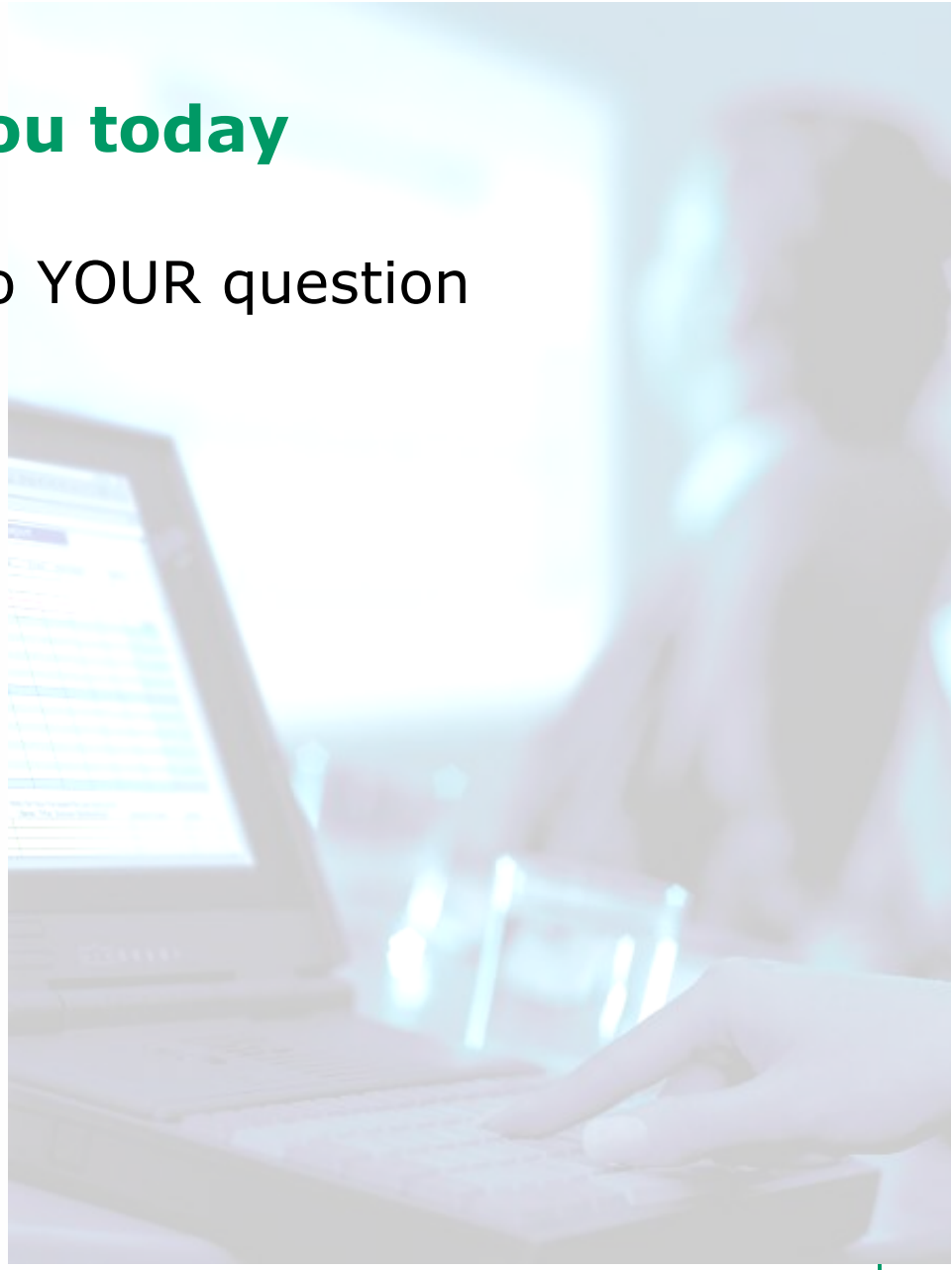
Access to experts, answers to YOUR question

Insider tips and tricks

Technical discussions

Connections with peers

Technical content, straight up.



What you'll hear about today

AMD
Smarter Choice



Application Optimisation
Multi-threading
Virtualization
Parallelization
Profiling
Monitoring
:

Reduced TCO
Lower power consumption
Scalability

We want to help you…

AMD
Smarter Choice

Optimize technology to create and maintain a competitive edge

Complement your software with technical interfaces and hardware for a total solution

Stay on top of technology and its future direction

Make good coding decisions

In short… **Find better ways to create a quality product faster**

AMD's Commitment to Developers

AMD
Smarter Choice

Tools and libraries

- AMD CodeAnalyst
- AMD Core Math Library
- AMD Performance Library
- AMD SimNow!
- Partner tools co-developed with AMD for maximum Optimisation

Remote development environment

- Access to fully configured AMD64 systems to develop, test, and optimize applications

AMD Developer Central free resources

- Documentation
- Training
- Technical articles
- Case studies
- Discount hardware

AMD Developer Days

- Local and customized for you

Turn in the attendee survey at the end of the day for your special gift.

AMD
Smarter Choice

Let's go!

Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice



AMD
Smarter Choice

AMD Architecture

Justin Boggs
Sr. Developer Relations Engineer

Agenda

AMD
Smarter Choice

- AMD Processors Overview
- AMD Technologies
- Microarchitecture
- Roadmap and Futures



AMD
Smarter Choice

AMD Processors Overview

AMD Processor Brands

AMD
Smarter Choice

Server/Workstation



Desktop









Mobile









Server/Workstation Brand Positioning

AMD
Smarter Choice

PERFORMANCE 4-WAY AND 8-WAY



AMD Opteron™ 800 Series & 8000 Series Processors

- Designed for 4-way and 8-way Server solutions
- First native x86 dual-core solution for 4-way / 8-way computing

PERFORMANCE 2-WAY



AMD Opteron™ 200 Series & 2000 Series Processors

- Designed for 2-way Server / Workstation solutions
- First native x86 dual-core solution for 2-way computing

PERFORMANCE 1-WAY



AMD Opteron™ 100 Series & 1000 Series Processors

- Designed for 1-way Server / Workstation solutions
- First native x86 dual-core solution for 1-way computing

AMD Opteron™ Processors for Servers and Workstations

- Direct Connect Architecture for great performance
- Industry leading performance-per-watt
- Designed with native quad-core solutions in mind



AMD
Smarter Choice

AMD Technologies

AMD Advantage

AMD
Smarter Choice

Built from the ground up, AMD64 processor technology makes it possible to improve responsiveness to changing business needs

AMD64 offers flexibility by supporting 32- and 64-bit applications on the same system and across desktop, mobile and server applications

Direct Connect Architecture improves overall system performance by removing the bottlenecks inherent in legacy front-side bus based architectures because everything is directly connected to the CPU

AMD's Native Multi-Core provides enhanced a better design and more performance without increasing power requirements

AMD PowerNow!™ Technology decreases overall power consumption by optimizing performance on demand

Enhanced Virus Protection adds an extra level of virus protection to your security solution

AMD64 Technology

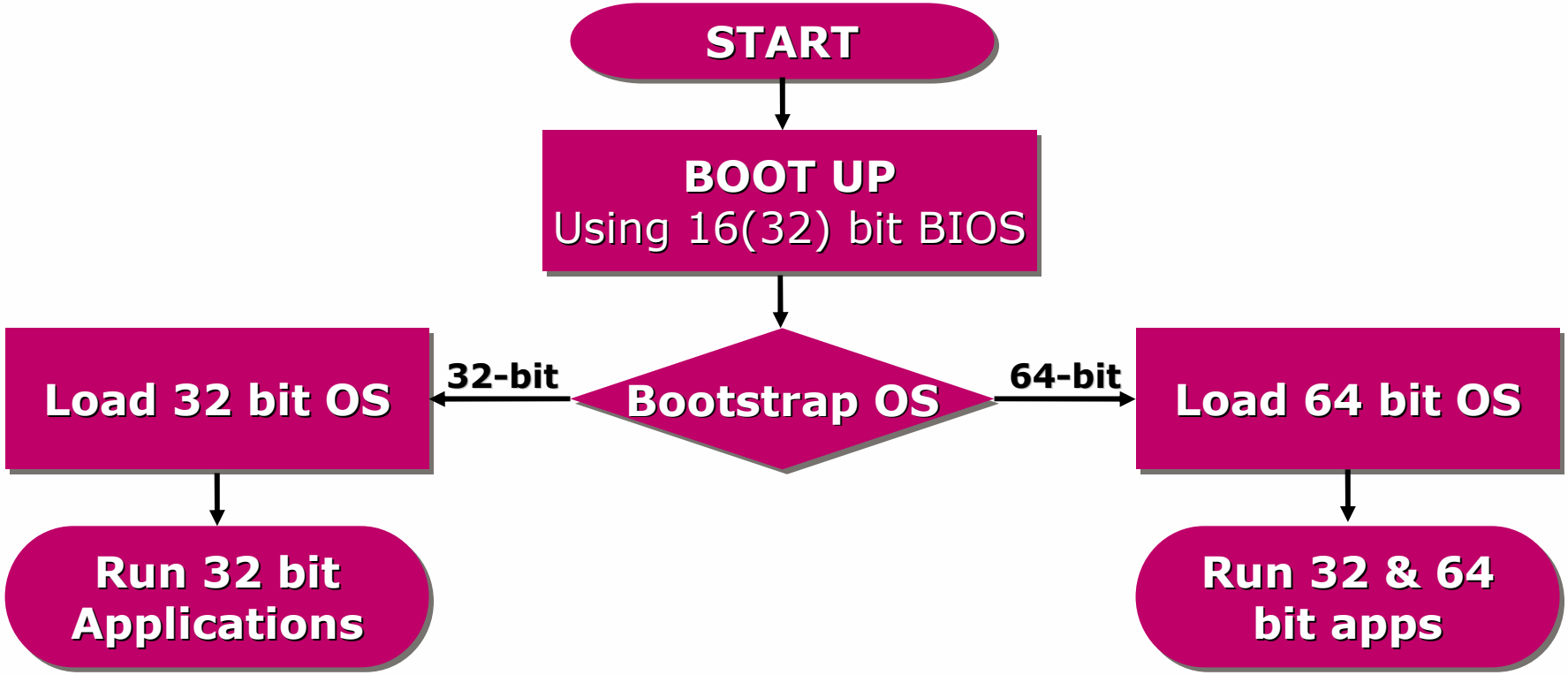
Powerful 64-Bit Computing

- Technology that gives total backward compatibility with leading-edge 32-bit computing performance.
- Technology that paves the way to multi-core computing with cutting-edge communications technology.
- Technology is more than 64-bit computing— it's also about next-generation architecture.
- Technology that allows software developers to create new functionality for end users.
- Technology that solves real problems.

AMD64 Technology: Compatibility

An AMD64 PC can run

Both 32- and 64-bit operating systems



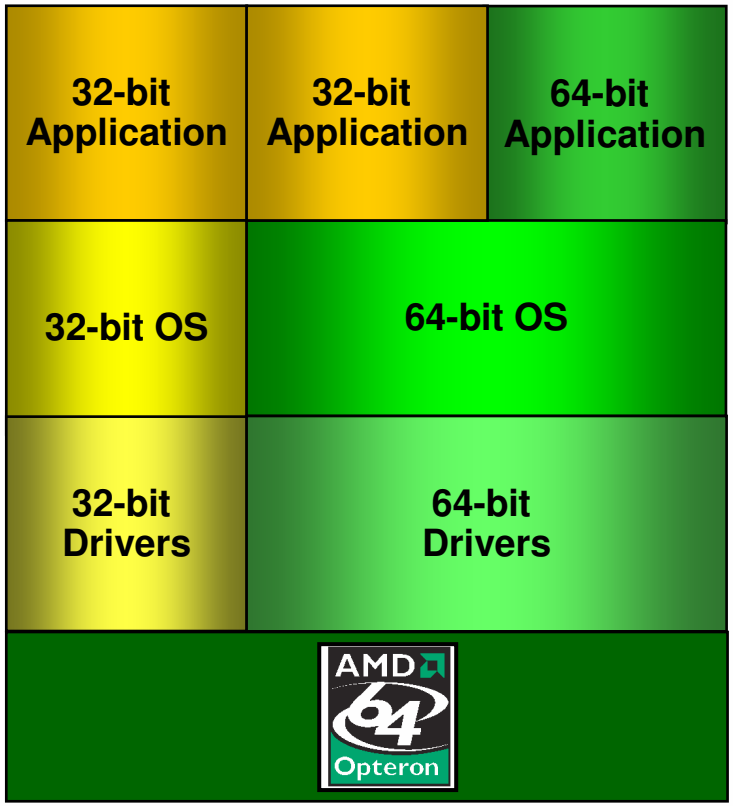
AMD64 Computing Strategy

AMD
Smarter Choice

Practical approach to introducing new capabilities

Extensions so simple and compatible, that the processor can support both x86 32-bit and AMD64 at full speed & performance

- **Compatibility of 32-bit applications retained at full speed, without decreasing performance.**
- **Can move to 64-bit addressing and data types without giving up 32-bit compatibility (long mode)**
- **64-bit applications can be developed using familiar techniques and tools (only two new instructions were added to support 64-bit addressing)**
- **Leverages the existing key PC infrastructure rather than needing to re-invent it.**

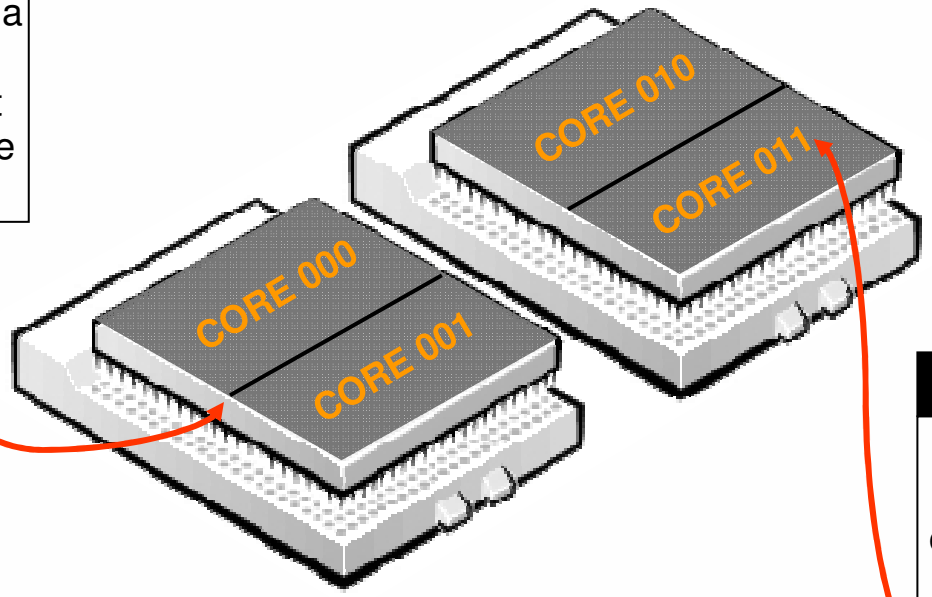


Processor vs. Core Terminology

AMD
Smarter Choice

Processor

Physical packaged die that plugs into a socket on the motherboard that contains 1 or more cores.

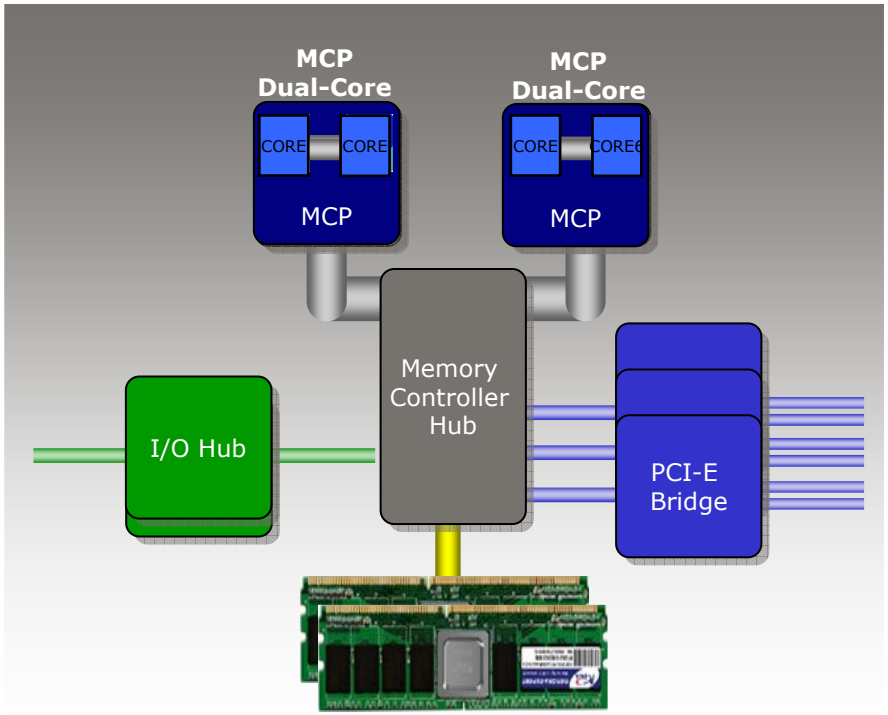


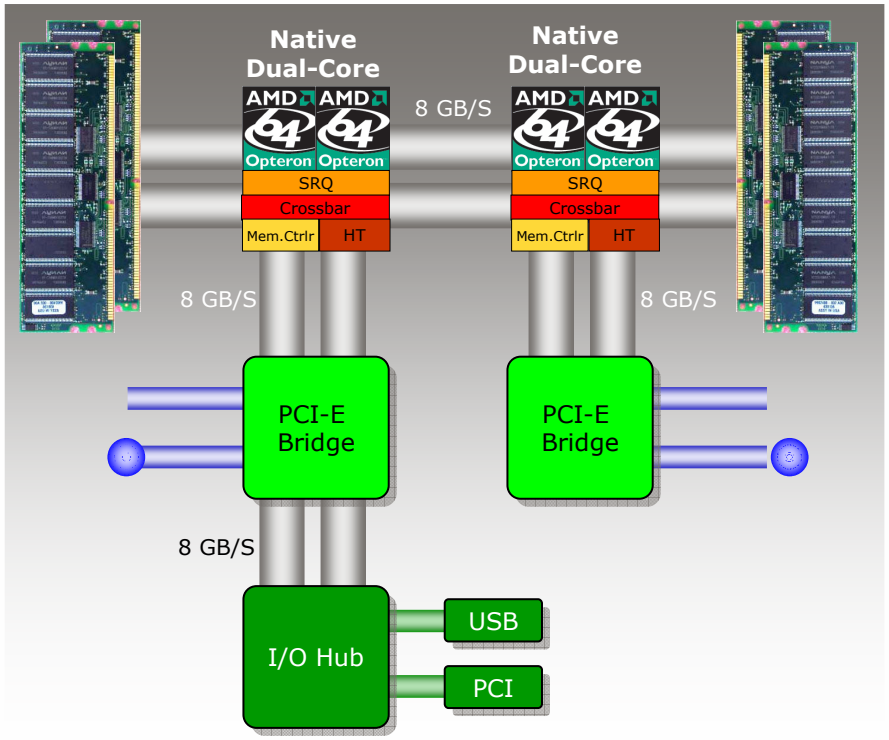
Core

1 complete private set of registers, execution units, and retirement queues needed to execute x86 programs; managed & scheduled as single x86 processing resource by the OS.

Direct Connect Architecture

Balanced Platform Bandwidth





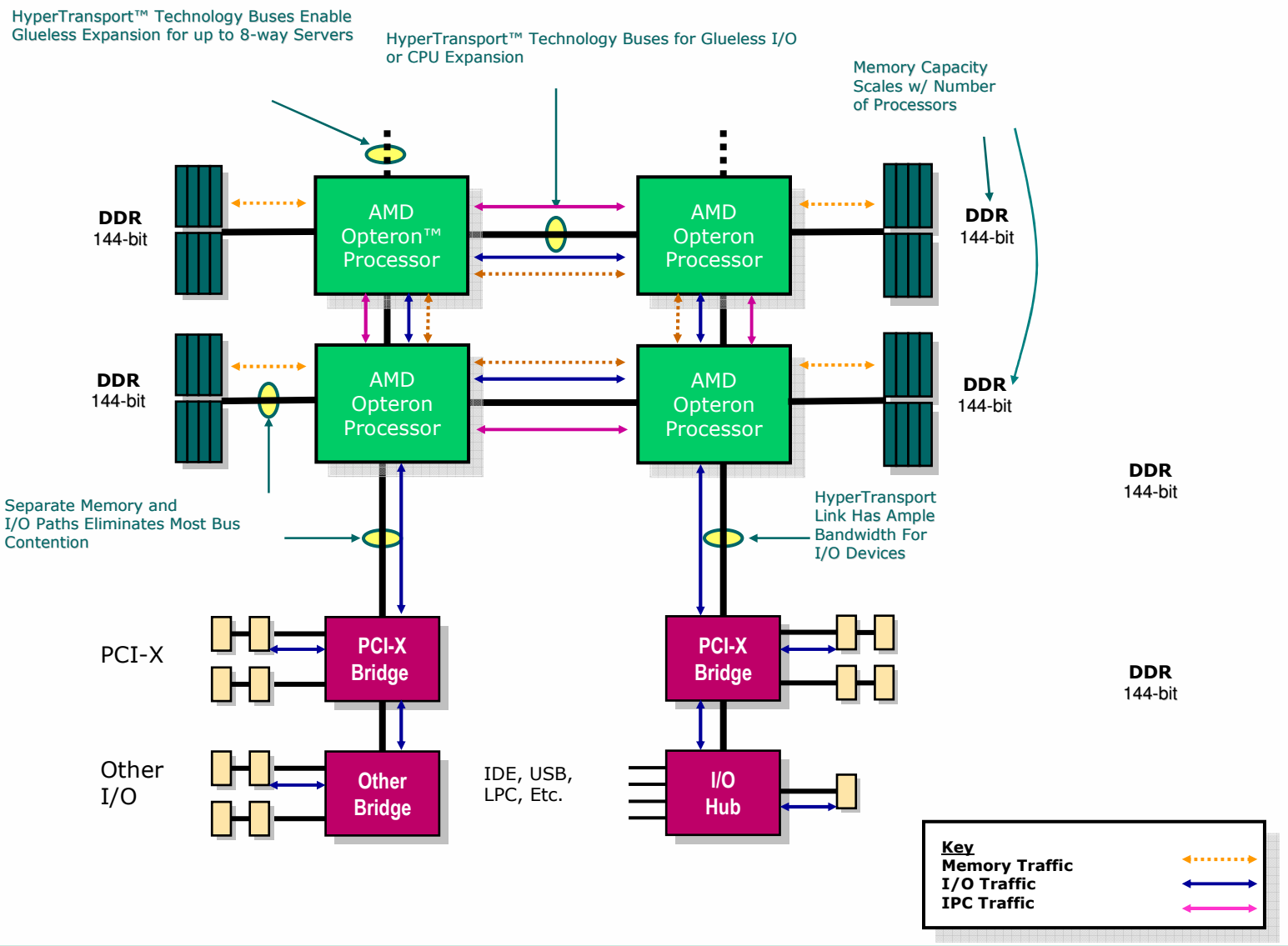
Legacy x86 Architecture

- 20-year old front-side bus architecture
- CPUs, Memory, I/O all share a bus
- Can be a bottleneck to performance
- Intel Xeon "Dempsey" & Intel Xeon "Woodcrest"

AMD64 Technology with Direct Connect Architecture

- Industry-standard AMD64 technology
- AMD's revolutionary Direct Connect Architecture helps eliminate bottlenecks inherent in traditional FSB architectures
- HyperTransport™ technology interconnect for high bandwidth and low latency

HyperTransport™

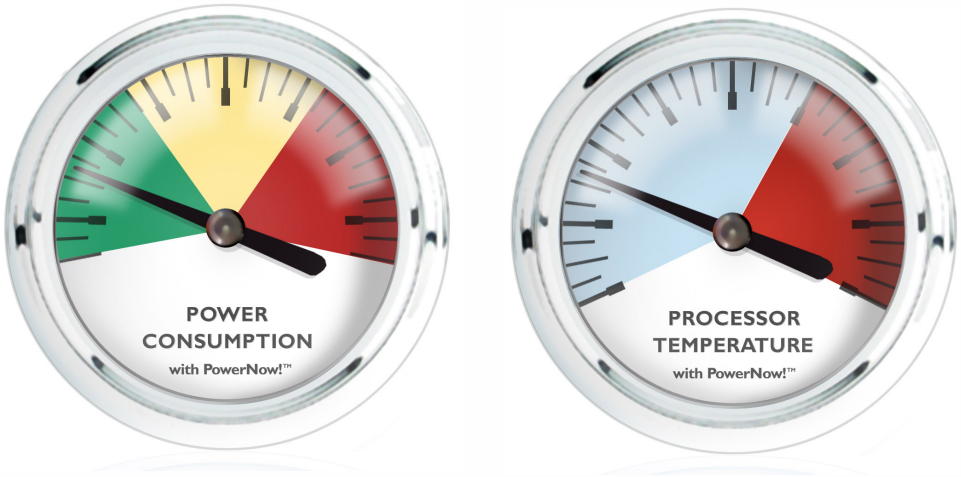


AMD PowerNow!™ Technology

With AMD's efficient core design,
AMD PowerNow! Technology optimally self-regulates power consumption
without performance interruptions or delays

AMD PowerNow! Technology

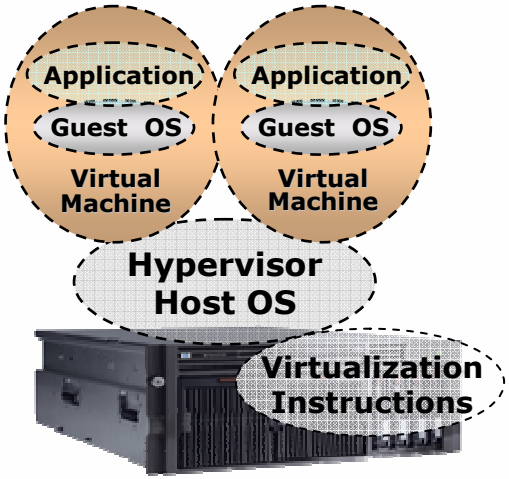
- Reduces power at idle by up to 65%
- Optimizes platform power consumption
- Provides performance on demand by dynamically adjusting performance based on CPU utilization



Next-Generation AMD Opteron™
AMD Virtualization™

AMD
Smarter Choice

Hardware-enabled **AMD Virtualization™** is designed to improve all aspects of x86-based virtualization



AMD Virtualization Benefits

- Instructions added to help **reduce complexity** of virtualization software
- **Reducing overhead** by selectively intercepting information destined for guest
- Enables Guest OS's to run unmodified for **easier implementation and support**
- Increased isolation to **improve security** of virtual machines
- Offers architectural enhancements to improve **efficiency of switching** between hypervisor and the guest OS's

Virtualization in the processor: SVM!

AMD Smarter Choice

- Native virtualization of x86 architecture requires "unnatural acts" to achieve – leading to increased performance overhead & complexity
- Moving functionality traditionally served by software-based hypervisor into the processor helps to solve these problems.
- ***SVM – Secure Virtual Machine is next logical evolution to AMD's Direct Connect Architecture* to provide technology for silicon enhanced virtualization**
- SVM allows the software vendors to focus on the value-add, leaving the worry of proper emulation to the processor.

SVM virtualization technology allows AMD to continue to offer a competitive performance roadmap while meeting the system architecture demands of our customers

SVM Architecture Features

AMD
Smarter Choice

New Instruction Set Features

- Processor Mode: Guest Mode
- Data Structure: Virtual Machine Control Block (VMCB), etc
- Instructions: VMRUN, VMLOAD, VMSAVE, VMMCALL, etc
- All instructions now Restartable

Interrupt architecture changes

- Selective Interception:
 - Increasing performance and enabling para-virtualization
- Event Injection
 - Eliminates need for VMM code to emulate x86 exception delivery
 - Designed to reduce VMM development time significantly

Security Features: DMA Exclusion Vectors, SKINIT, etc

Core SVM Architecture: VMRUN

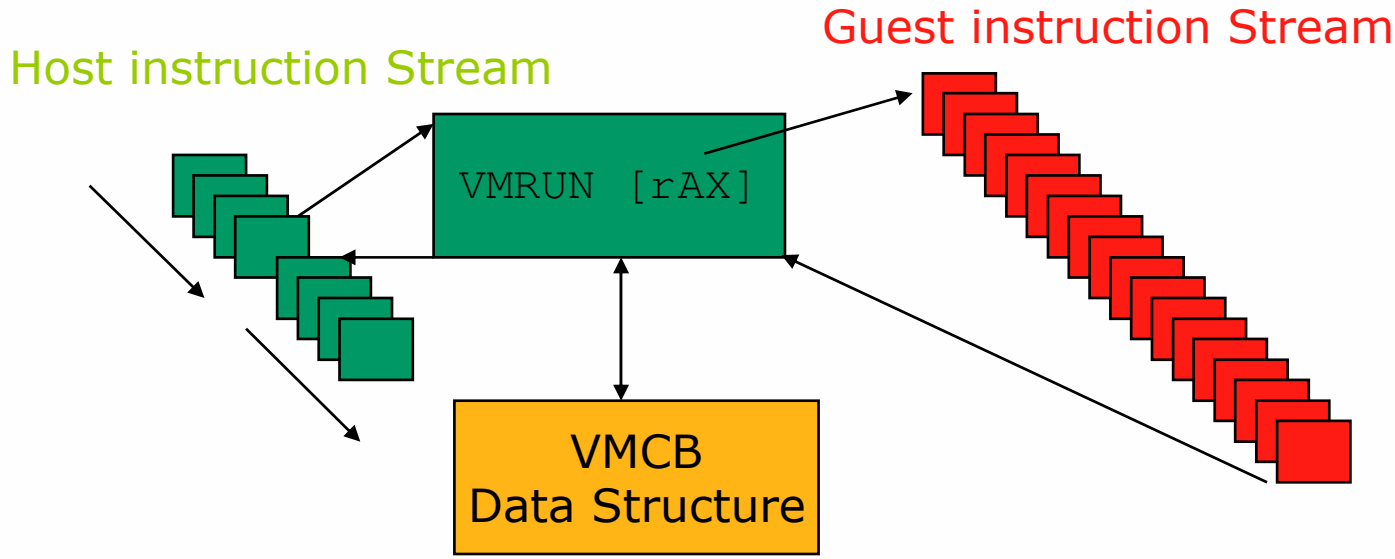
Virtualization based on Virtual Machine Run (VMRUN) instruction

VMRUN executed by host causes the guest to run

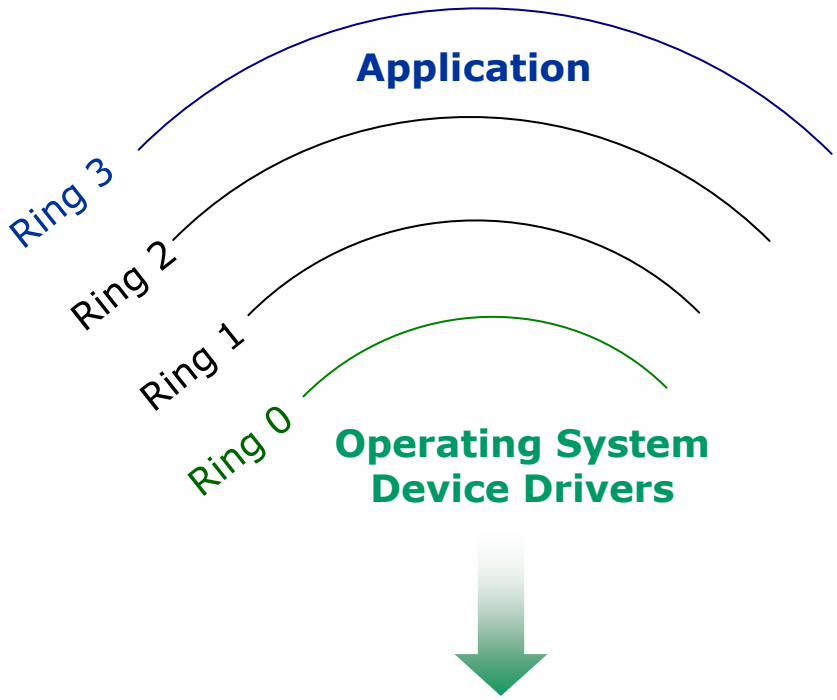
Guest runs until it exits back to the host

World-switch: host → guest → host

Host resumes at the instruction following VMRUN



Virtualization In An x86 World

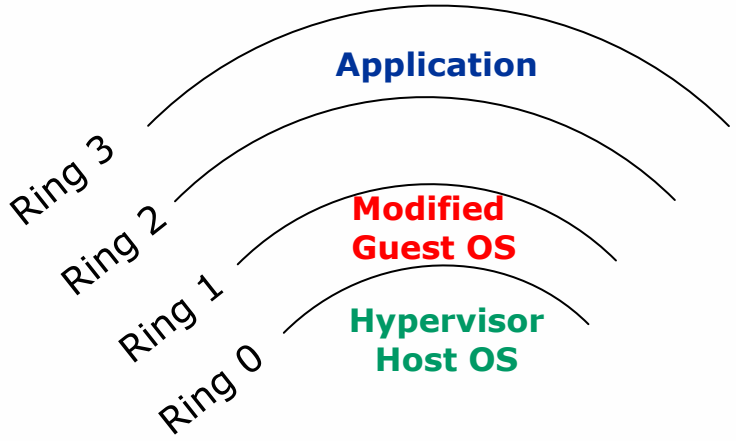


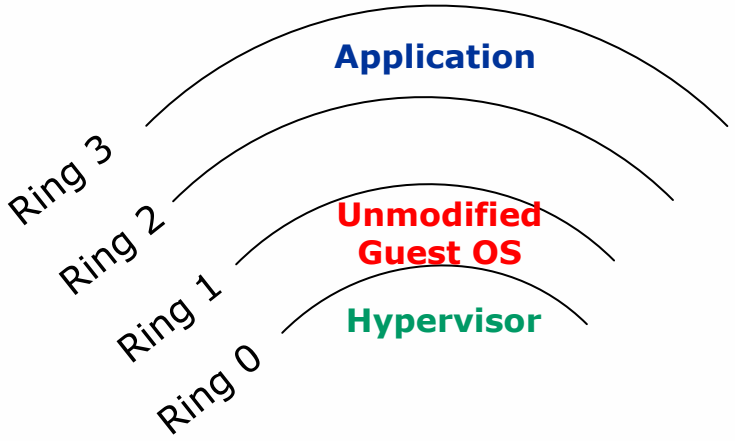
Only privileged code can define memory-mapping parameters or set up interrupt handlers

- x86 architecture supports different classes of privileged operations, often called rings
- Software running in a lower-numbered ring can exercise control over software running at a higher-numbered ring
- With x86 architecture, the OS expects direct control of the CPU (runs at "Ring O")
- Traditional x86 processors can't simultaneously run both host and guest OS's at Ring 0

Virtualization In An x86 World

Software-Enabled Virtualization





- **Para-virtualization**
- Hypervisor handles Host/Guest OS interactions
- Guest OS's limited to open source OS's (does not support Microsoft OS's)
- Guest OS is recompiled to run at Ring 1
- Host OS handles devices
- Example: Xen

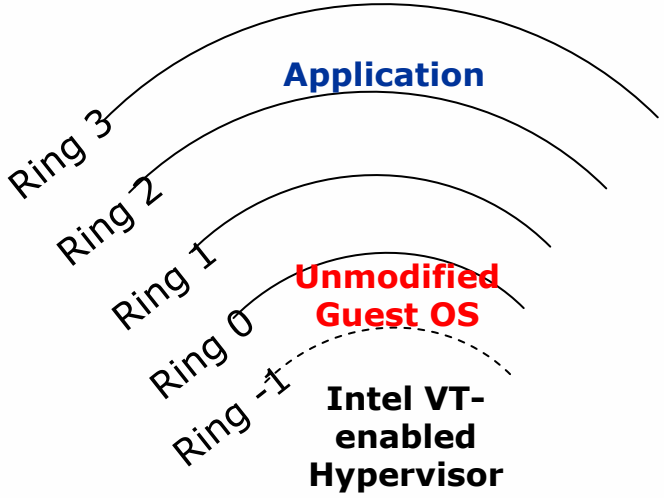
Full Virtualization (Binary Emulation)

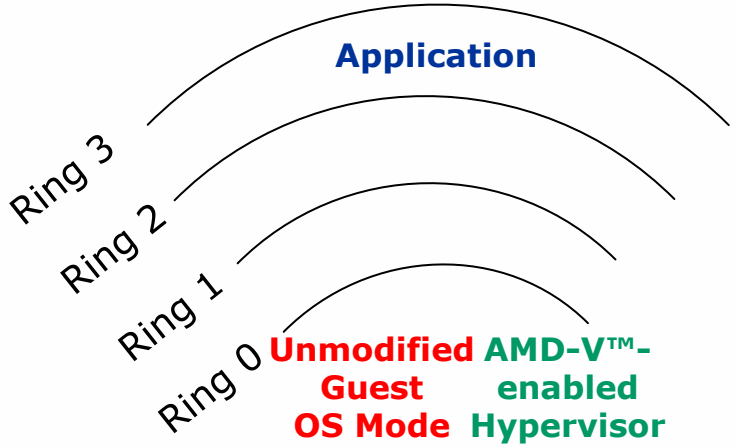
- Hypervisor is also Host OS
- Intercepts and emulates instructions and virtualizes devices
- Guest OS runs unmodified
- Example: VMware ESX

AMD Virtualization™ versus Intel VT

AMD Smarter Choice

Hardware-Enabled Virtualization





Intel VT

- Creates a new privilege layer for running Hypervisors
- Hypervisor controls access to devices
- Untagged TLB requires more memory accesses to switch between host/guests
- Off-chip memory controller requires Hypervisor to take on more of memory management work

AMD Virtualization™ Technology

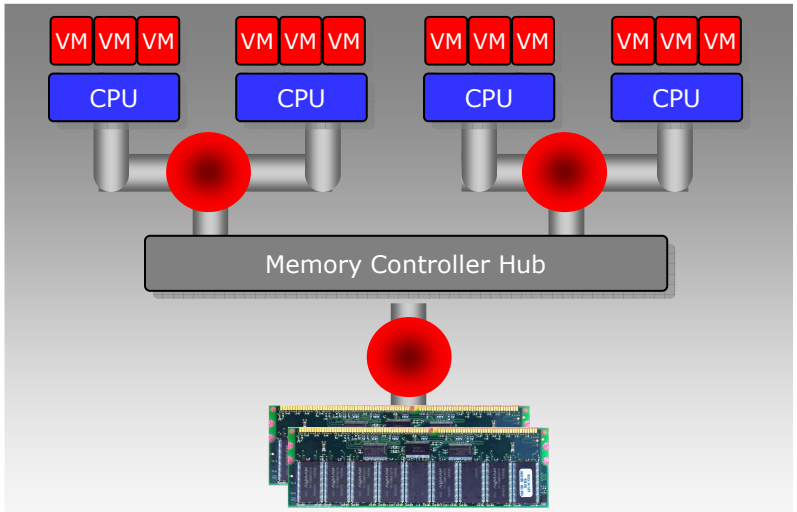
- Creates a Guest Mode that allows Guest OS to run unmodified with Hypervisor
- Hypervisor controls access to devices
- Tagged TLB reduces the number of memory accesses needed to switch between host/guests
- Upcoming Nested Page Tables for additional performance tuning

AMD Virtualization™ versus Intel VT

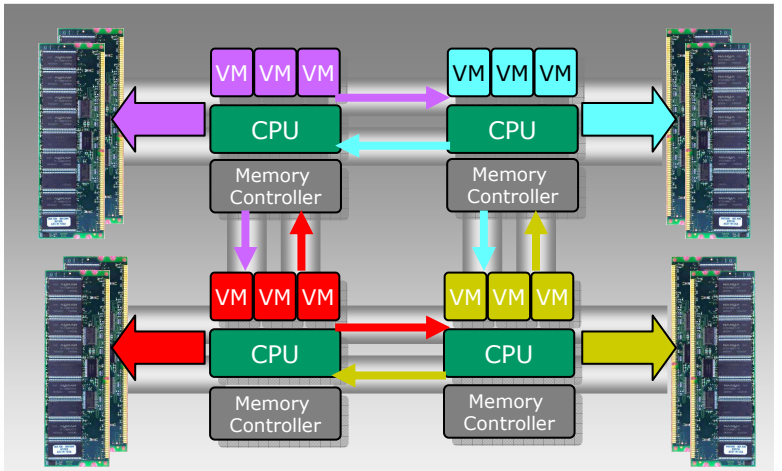
- The two most fundamental needs of virtualized environments are **security** and **performance**
- The key differences between AMD-V™ and Intel VT lie in **how efficiently we address these needs**

	AMD Approach	Intel Approach
Security	Hardware-based **DEV** enforces security between VMs with less overhead than software methods	Slower, less-efficient software-based security
Performance	**Direct Connect Architecture** provides superior memory and CPU scaling	Front-side bus based architecture delays communication between memory and CPU(s)
	Tagged TLBs reduce memory accesses required to load a new VM (no TLB flush is required), improving performance when switching between VMs	Must flush TLB buffers each time control passes from one VM to another, reducing efficiency of switching
	Nested Page Tables (planned for 2007) will further improve efficiency of switching between VMs	(no comparable feature)

AMD Virtualization™ versus Intel VT



Shared memory can create bottlenecks

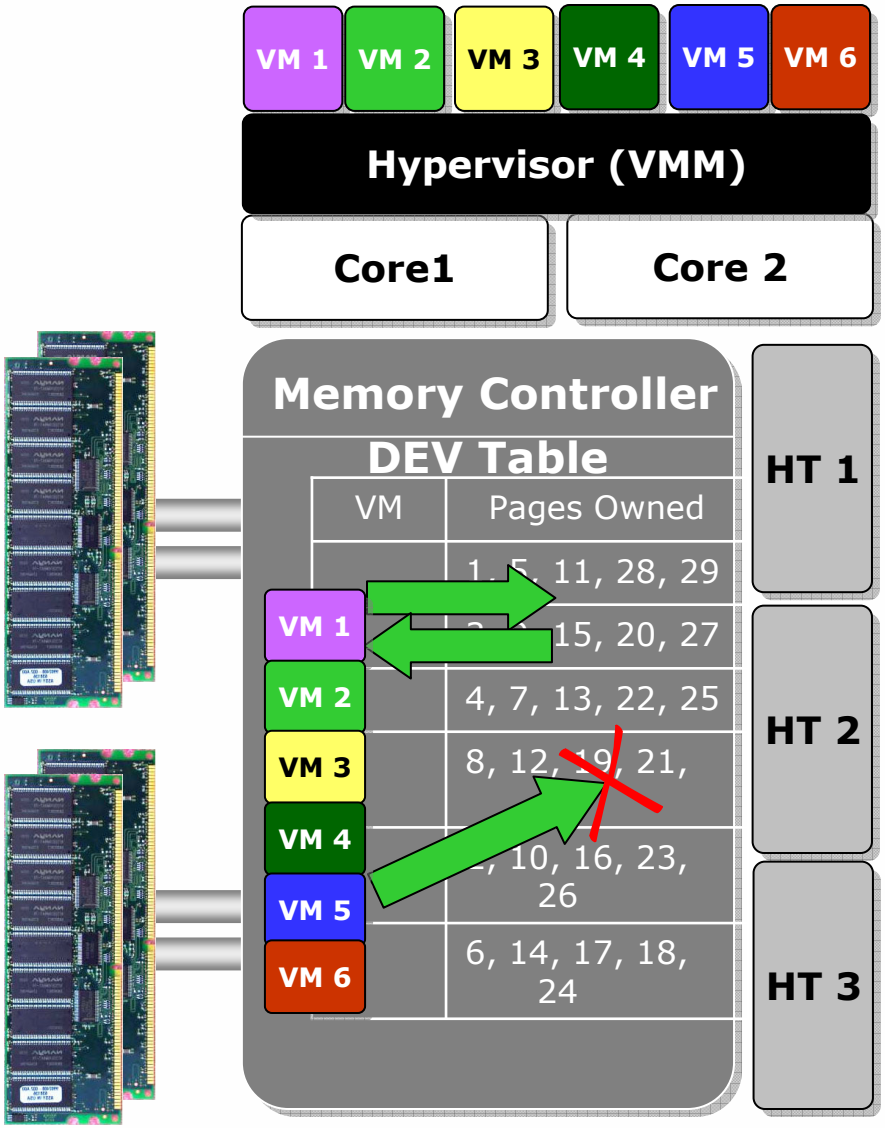


Dedicated memory for scalability

- **Shared front-side bus** can decrease application performance *within* a virtual machine
- **Untagged TLB** means less efficient switching *between* virtual machines
- Software-based memory management and security (via **external Memory Controller Hub**) can reduce overall virtualization performance and efficiency

- **Direct Connect Architecture** helps improve application performance *within* a virtual machine
- **Tagged TLB** means more efficient switching *between* virtual machines
- Hardware-based memory management and security (**Integrated memory controller with DEV**) can improve overall virtualization performance and efficiency

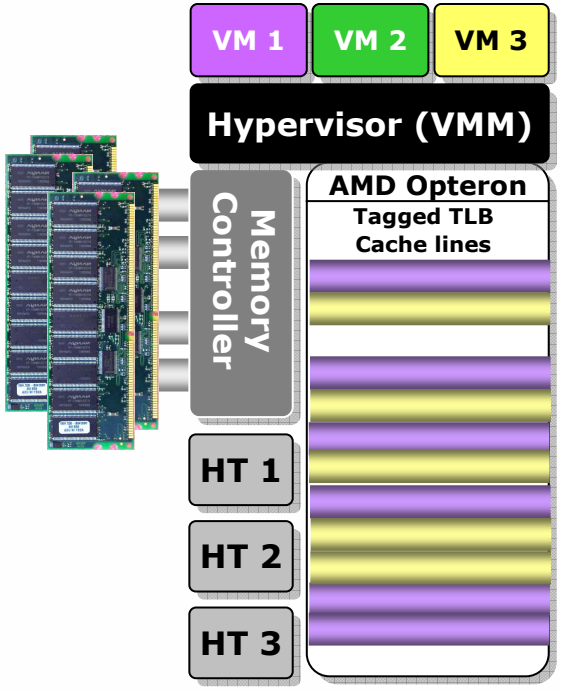
Device Exclusion Vector (DEV)



- DEV lets the Hypervisor (VMM) know if a device is allowed to access a page of memory or not
- So DEV **improves virtualization security** by denying memory accesses for unauthorized requests
- For example:
 - VM 1 Requests page 28, access is **granted** … quickly
 - VM 5 Requests page 25, access is **denied** … quickly
- **Xeon can do this**, but it happens in software … so **it happens slower**
- Only processors with an **Integrated Memory Controller** offer this benefit

Tagged Translation Look-aside Buffer

- TLB is a table in the CPU that contains cross-references between the virtual and real addresses of recently referenced pages of memory
- "Tagged" means the CPU knows which data belong to a virtual machine

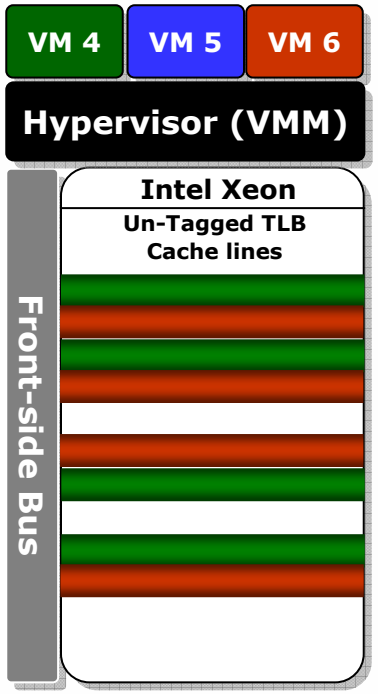


- So, for example:

VM 1 runs on the CPU and loads additional data from memory

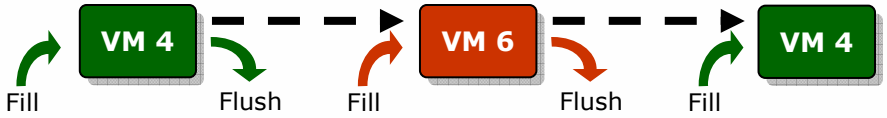
VM 3 takes control and loads its data, other TLB data remains

VM 1 takes control back the data it needs is there … resulting in **better performance**



- Xeon's VMM must flush the TLB every switch, **hurting performance**

VMM Control:



AMD Is Driving x86 Virtualization

AMD Smarter Choice

AMD64 Technology Road Map

Direct Connect Architecture	Virtualization Extensions (Today)	Nested Page Tables (2007*)	I/O Virtualization (2008*)
Increase performance with multi-core capabilities and fast memory access	**Simplify virtualization software and helps reduce CPU overhead**	**Reduce overhead of switching between virtual machines**	**Virtualize devices to improve performance and security**

Xen Novell Microsoft vmware redhat SOLARIS Xen Source Virtual Iron egenera SWsoft Automation and Virtualization

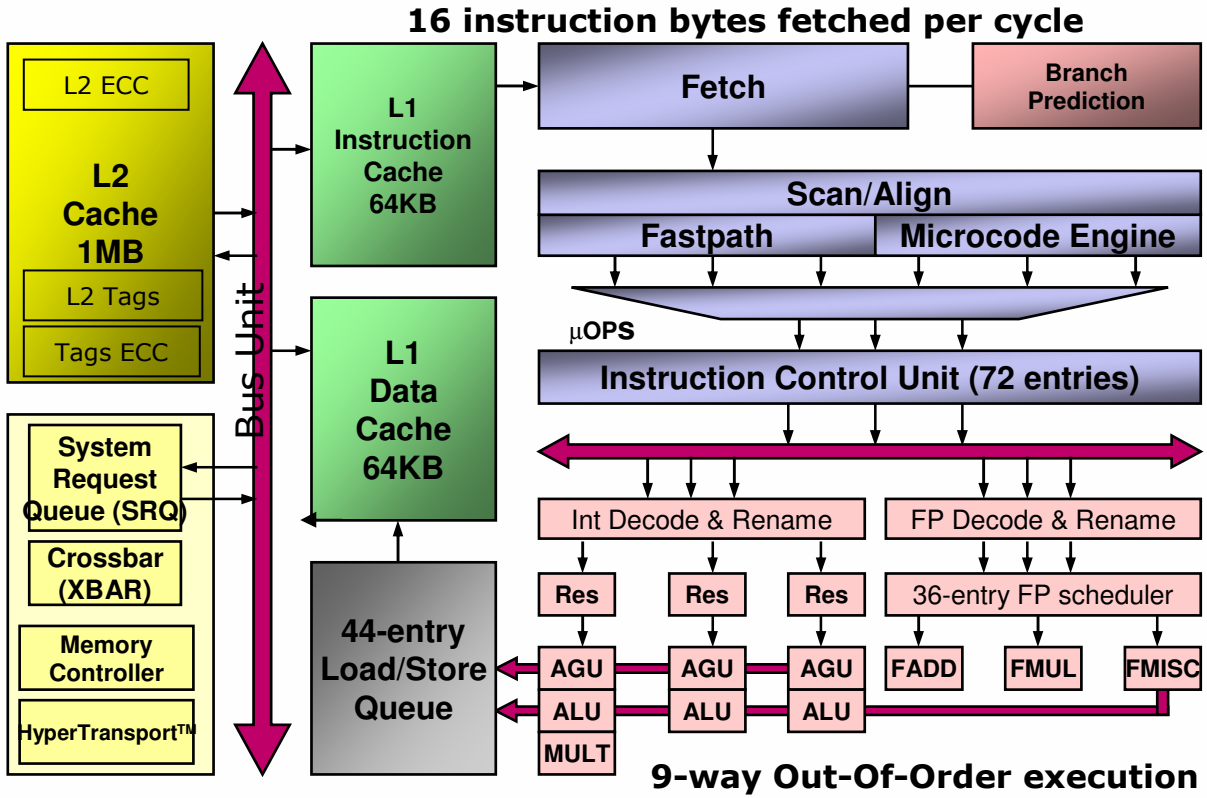
- **AMD is developing a robust virtualization ecosystem that provides users with innovation and choice**



AMD
Smarter Choice

Microarchitecture

AMD Opteron™ Processor Core Facts



12-stage Int, 17-stage fast-path pipelines

Enhanced TLB structures w/flush filter

Opts for off-loading writes, probes, memory

16 SSE & SSE2 128-bit xmm registers

8 legacy x87 80-bit registers

FPU Throughput

36 entry FPU instruction scheduler

64-bit/80-bit FP Realized thru-put: 1 Mul + 1 Add per cycle

32-bit FP Realized thru-put: 2 Mul + 2 Add per cycle

FP & Integer Throughput

AMD Smarter Choice

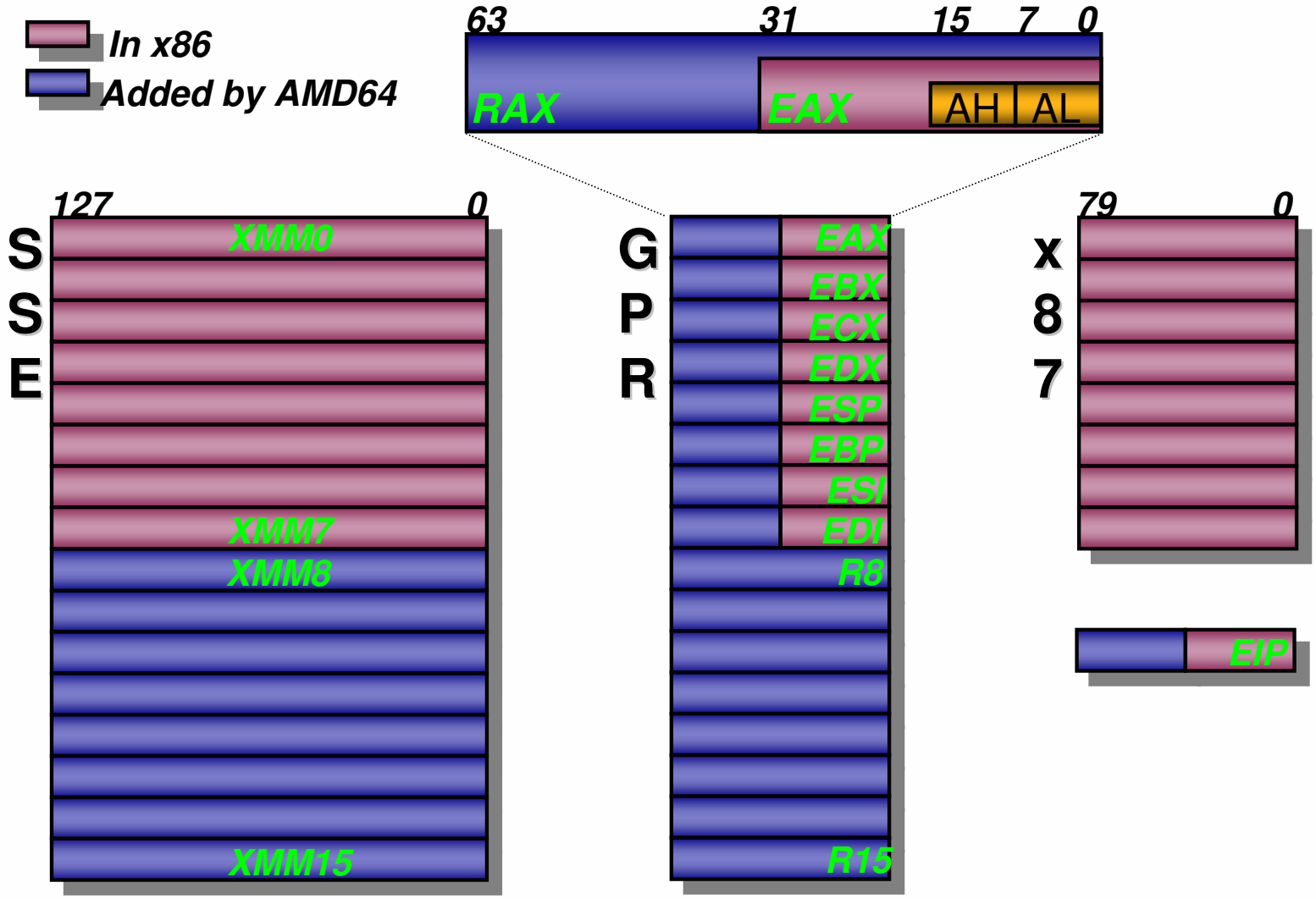
- **FPU Throughput**
 - SSE2, x87
 - ✓ Theoretical: (1 Mul + 1 Add)/cycle
 - SSE, 3DNow!
 - ✓ Theoretical: (2 Mul + 2 Add)/cycle

- **32-bit Integer Throughput**
 - 3 add / clock cycle
 - 1 multiply + 2 adds / clock cycle
 - Multiply latency has shrunk from 5 cycles on AMD Athlon™ 64 to 3 cycles on the AMD Opteron™

- **64-bit Integer Throughput**
 - 1 add / clock cycle
 - 1 multiply every other clock cycle
 - Multiply latency is 4 cycles
 - Integer Instruction Scheduler
 - ✓ Out Of Order (OOO) from a queue of 24* Integer Macro-Ops

**AMD Athlon™ 64 Instruction Scheduler is 18 Macro-Ops deep*

The AMD64 Programmer's Model



AMD64 Instruction Set Architecture

AMD
Smarter Choice

Support for all x86 instruction extensions

- MMX™, SSE, SSE2, SSE3, 3DNow!™

Full performance with all code

- Native 32-bit x86 mode
- Enhanced capability in 64-bit mode
 - 64-bit general purpose registers
 - 64-bit addressing
 - Twice as many general purpose registers
 - Twice as many SSE registers

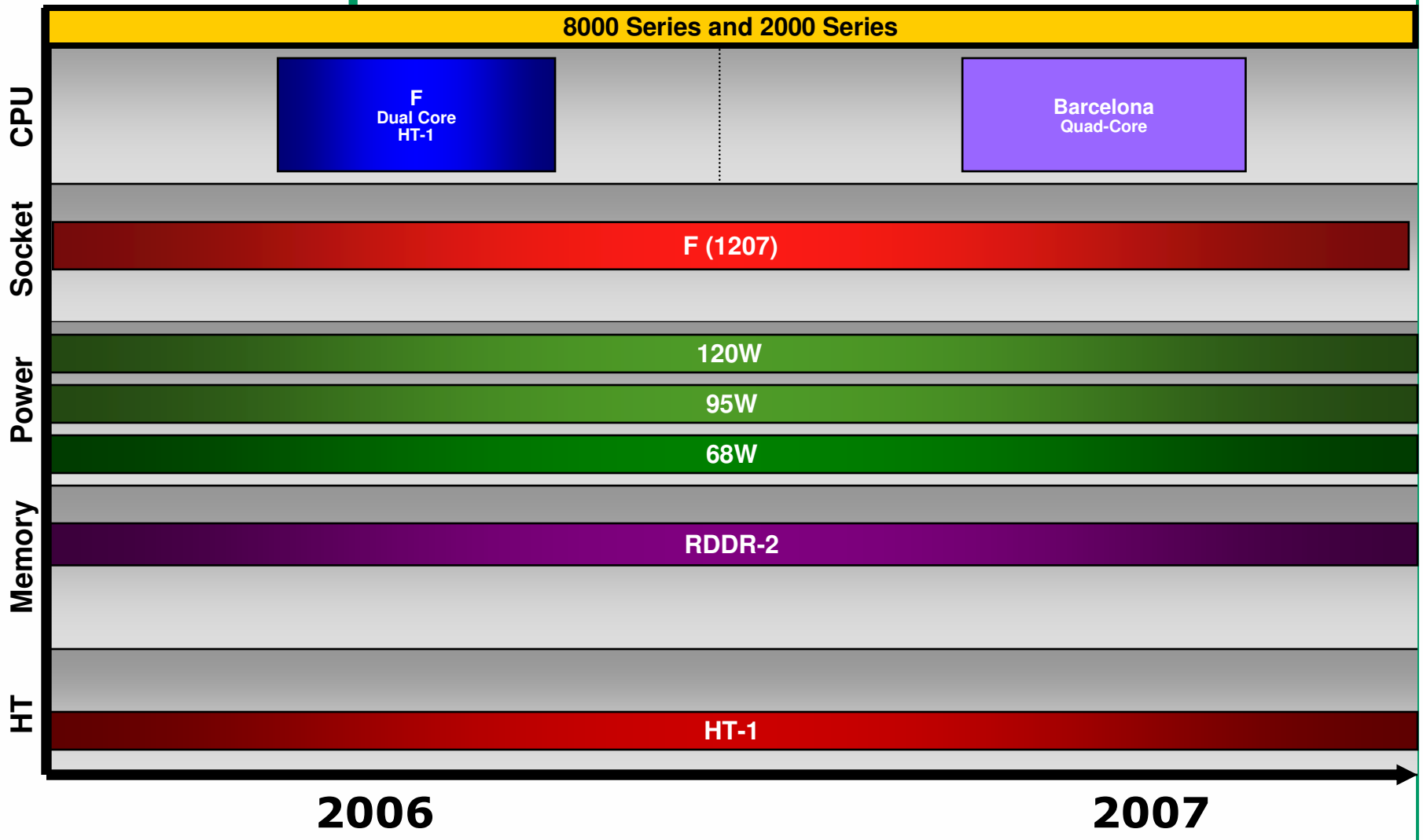
Same familiar x86 instructions



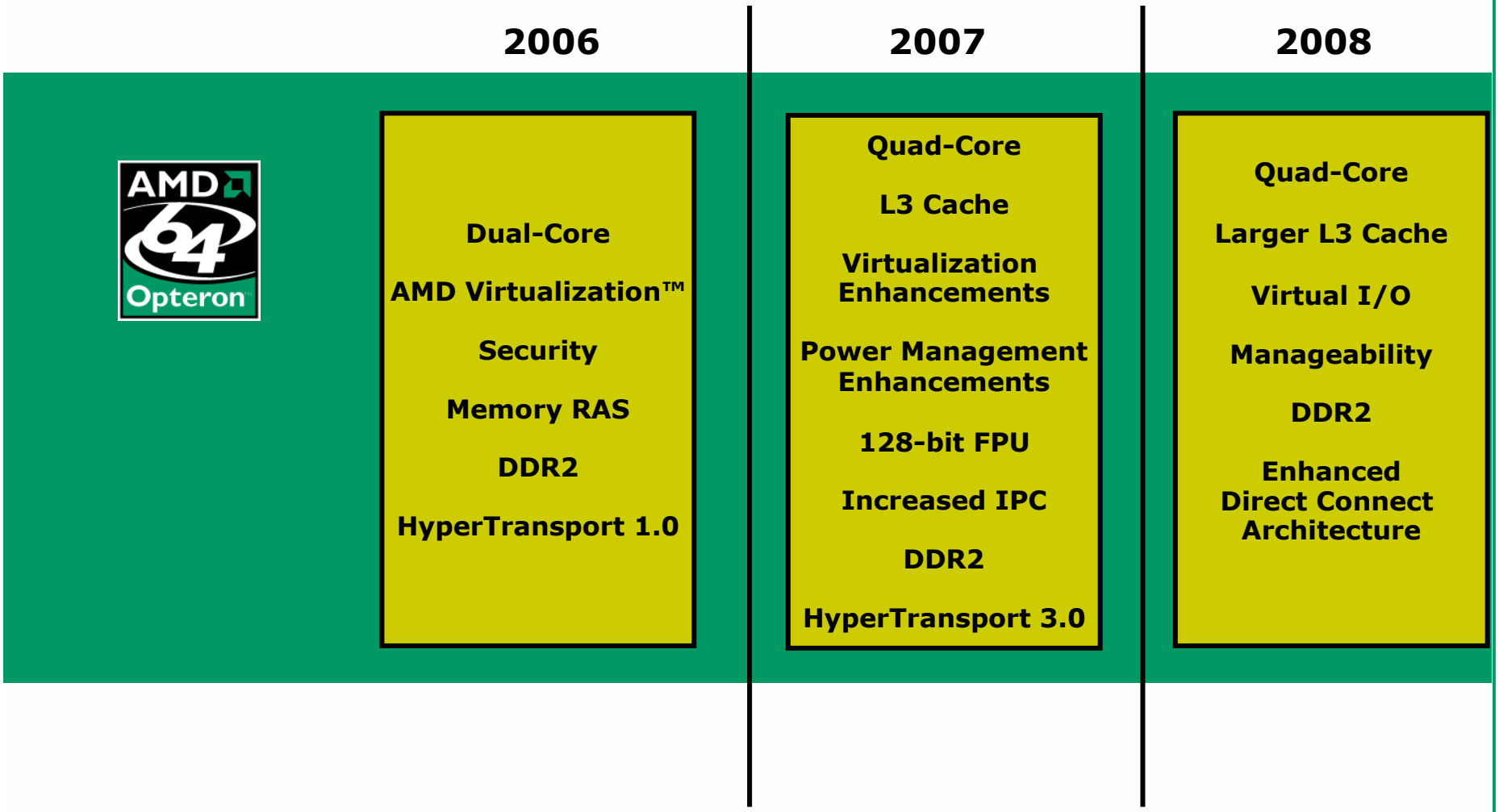
AMD
Smarter Choice

Roadmap & Features

AMD Opteron™ Processor Technology Roadmap:



Technologies Roadmap: Server/Workstation



Quad-Core AMD Opteron™
Continued Performance-Per-Watt Leadership

AMD Smarter Choice

Advanced Process Technology

Silicon-on-Insulator Process

Fast transistors with low power leakage - **helps reduce wasted power and heat**

Native Quad-Core Design

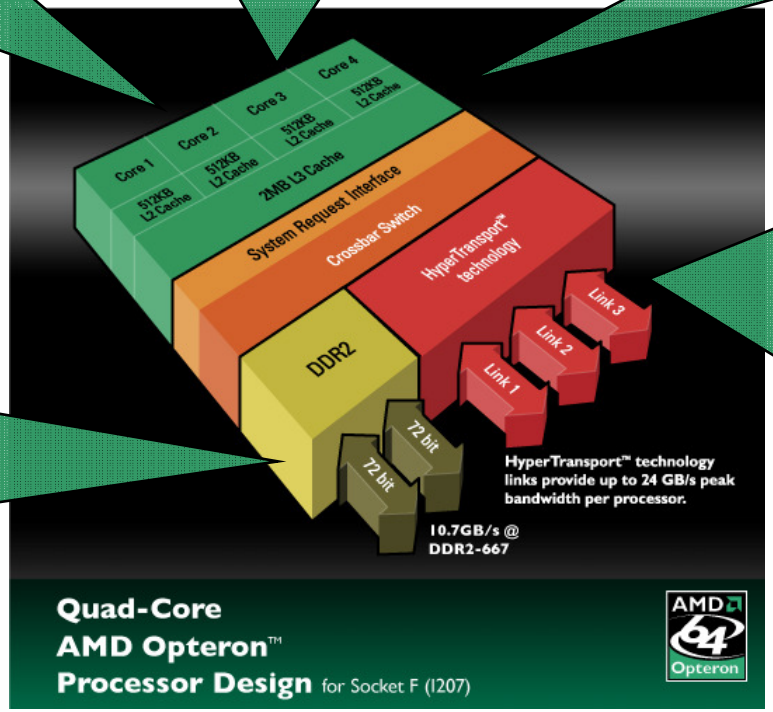
To increase performance-per-watt efficiencies **using the same thermal design power**

Enhanced AMD PowerNow!™ Technology with Optimized Power Management

Dynamically **reduces power used by cores** based on workload **for an up to 75% savings in power** per processor at CPU idle!

Integrated Memory Controller

Lower Power consumption
Integrated memory controller power included in processor power budget vs. **22 watts to ~48watts required by external memory controllers**!



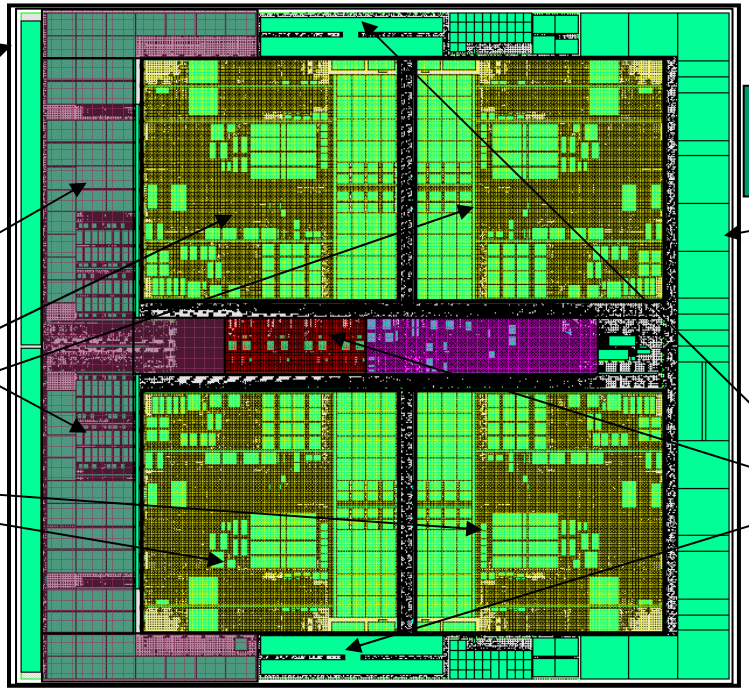
Direct Connect Architecture

- Provides **fast I/O throughput**
 - I/O directly connected
- Provides **faster CPU-to-CPU communication**
 - CPUs directly connected
- Integrated memory controller designed for **reduced memory latency and increased performance**
 - Memory directly connected

HELPS REDUCE TRADITIONAL x86 ARCHITECTURE FRONT-SIDE BUS BOTTLENECKS

A Closer Look at AMD's Quad-Core Server and Workstation Processor Architecture

AMD Smarter Choice



True quad-core die

Expandable shared L3 cache

IPC enhanced CPU cores

Optimum for 65nm SOI and beyond

Enhanced Direct Connect Architecture and Northbridge

- 32-bit instruction fetch
- Enhanced branch prediction
- Out-of-order load execution
- Dual 128-bit SSE dataflow
- Up to 4 dual precision FP OPS/cycle
- Dual 128-bit loads per cycle
- Bit Manipulation extensions (LZCNT/POPCNT)
- SSE extensions (EXTRQ/INSERTQ, MOVNTSD/MOVNTSS)

- HyperTransport links (Up to 8GB/sec)
- Enhanced crossbar design
- Enhanced AMD PowerNow!™ Technology with Optimized Power Management
- DDR2 support
- Enhanced RAS

SSE 128 – What's in it for the Customer?

Commercial and Consumer Benefits

AMD
Smarter Choice

High Performance Technical Computing	Media Encode and Decode
Oil and Gas Sims EDA Financial Analysis BioScience	HD Video Audio DVD Authoring

GOAL:
Double vector SSE performance
Both SSE Floating-point and SSE Packed Integer
Avoid creating bottlenecks in instruction or data delivery

Comprehensive Upgrades for SSE128

Current Generation versus Next Generation

AMD
Smarter Choice

Parameter	Current Processor	"Barcelona"
SSE Exec Width	64	128 + SSE MOVs
Instruction Fetch Bandwidth	16 bytes/cycle	32 bytes/cycle + Unaligned Ld-Ops
Data Cache Bandwidth	2 x 64bit loads/cycle	2 x 128bit loads/cycle
L2/NB Bandwidth	64 bits/cycle	128 bits/cycle
FP Scheduler Depth	36 Dedicated x 64-bit ops	36 Dedicated x 128-bit ops

Can perform SSE MOVs in the FP "store" pipe

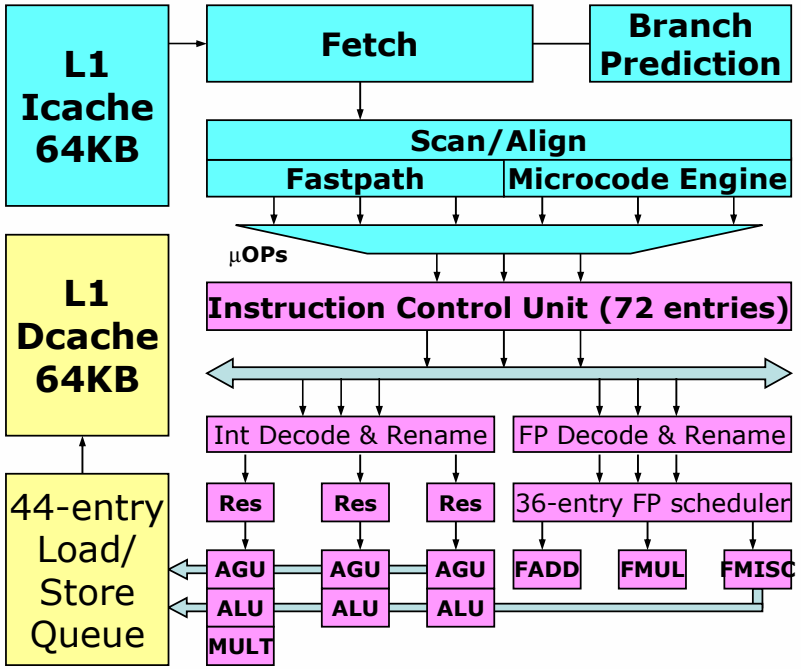
- Execute two generic SSE ops + SSE MOV each cycle (+ two 128-bit SSE loads)

SSE Unaligned Load-Execute mode

- Remove alignment requirements for SSE ld-op instructions
- Eliminate awkward pairs of separate load and compute instructions
- *To improve instruction packing and decoding efficiency*

CPU Core IPC Enhancements

- Advanced branch prediction
- 32B instruction fetch
- Sideband Stack Optimizer
- Out-of-order load execution
- TLB Optimisations
- Data-dependent divide latency
- More Fastpath instructions
 - CALL and RET-Imm instructions
 - Data movement between FP & INT
- Bit Manipulation extensions
 - LZCNT/POPCNT
- SSE extensions
 - EXTRQ/INSERTQ,
 - MOVNTSD/MOVNTSS



CPU Core IPC Enhancements

AMD Smarter Choice

- **Advanced branch prediction**
- 32B instruction fetch
- Sideband Stack Optimizer
- Out-of-order load execution
- TLB Optimisations
- Data-dependent divide latency
- More Fastpath instructions
 - CALL and RET-Imm instructions
 - Data movement between FP & INT
- Bit Manipulation extensions
 - LZCNT/POPCNT
- SSE extensions
 - EXTRQ/INSERTQ,
 - MOVNTSD/MOVNTSS

- **Dedicated 512-entry Indirect Predictor**
- **Double return stack size**
- **More branch history bits and improved branch hashing**

CPU Core IPC Enhancements

AMD Smarter Choice

- Advanced branch prediction
- **32B instruction fetch**
- Sideband Stack Optimizer
- Out-of-order load execution
- TLB Optimisations
- Data-dependent divide latency
- More Fastpath instructions
 - CALL and RET-Imm instructions
 - Data movement between FP & INT
- Bit Manipulation extensions
 - LZCNT/POPCNT
- SSE extensions
 - EXTRQ/INSERTQ,
 - MOVNTSD/MOVNTSS

- **Benefits integer code too**
- **Reduced split-fetch instruction cases**

CPU Core IPC Enhancements

AMD Smarter Choice

- Advanced branch prediction
- 32B instruction fetch
- **Sideband Stack Optimizer**
- Out-of-order load execution
- TLB Optimisations
- Data-dependent divide latency
- More Fastpath instructions
 - CALL and RET-Imm instructions
 - Data movement between FP & INT
- Bit Manipulation extensions
 - LZCNT/POPCNT
- SSE extensions
 - EXTRQ/INSERTQ,
 - MOVNTSD/MOVNTSS

- **Perform stack adjustments for PUSH/POP operations "on the side"**
- **Stack adjustments don't occupy functional unit bandwidth**
- **Breaks serial dependence chains for consecutive PUSH/POPs**

CPU Core IPC Enhancements

- Advanced branch prediction
- 32B instruction fetch
- Sideband Stack Optimizer
- **Out-of-order load execution**
- TLB Optimisations
- Data-dependent divide latency
- More Fastpath instructions
 - CALL and RET-Imm instructions
 - Data movement between FP & INT
- Bit Manipulation extensions
 - LZCNT/POPCNT
- SSE extensions
 - EXTRQ/INSERTQ,
 - MOVNTSD/MOVNTSS

- **New technology allows load instructions to bypass:**
 - **Other loads**
 - **Other stores which are known not to alias with the load**
- **Significantly mitigates L2 cache latency**

CPU Core IPC Enhancements

AMD Smarter Choice

- Advanced branch prediction
- 32B instruction fetch
- Sideband Stack Optimizer
- Out-of-order load execution
- **TLB Optimisations**
- Data-dependent divide latency
- More Fastpath instructions
 - CALL and RET-Imm instructions
 - Data movement between FP & INT
- Bit Manipulation extensions
 - LZCNT/POPCNT
- SSE extensions
 - EXTRQ/INSERTQ,
 - MOVNTSD/MOVNTSS

- **Support for 1G pages**
- **48bit physical address**
- **Larger TLBs key for:**
 - **Virtualized workloads**
 - **Large-footprint databases and transaction processing**
- **DTLB**
 - **Fully-associative 48-way TLB (4K, 2M, 1G)**
 - **Backed by L2 TLBs:**
 - ***512 x 4K, 128 x 2M***
- **ITLB**
 - **16 x 2M entries**

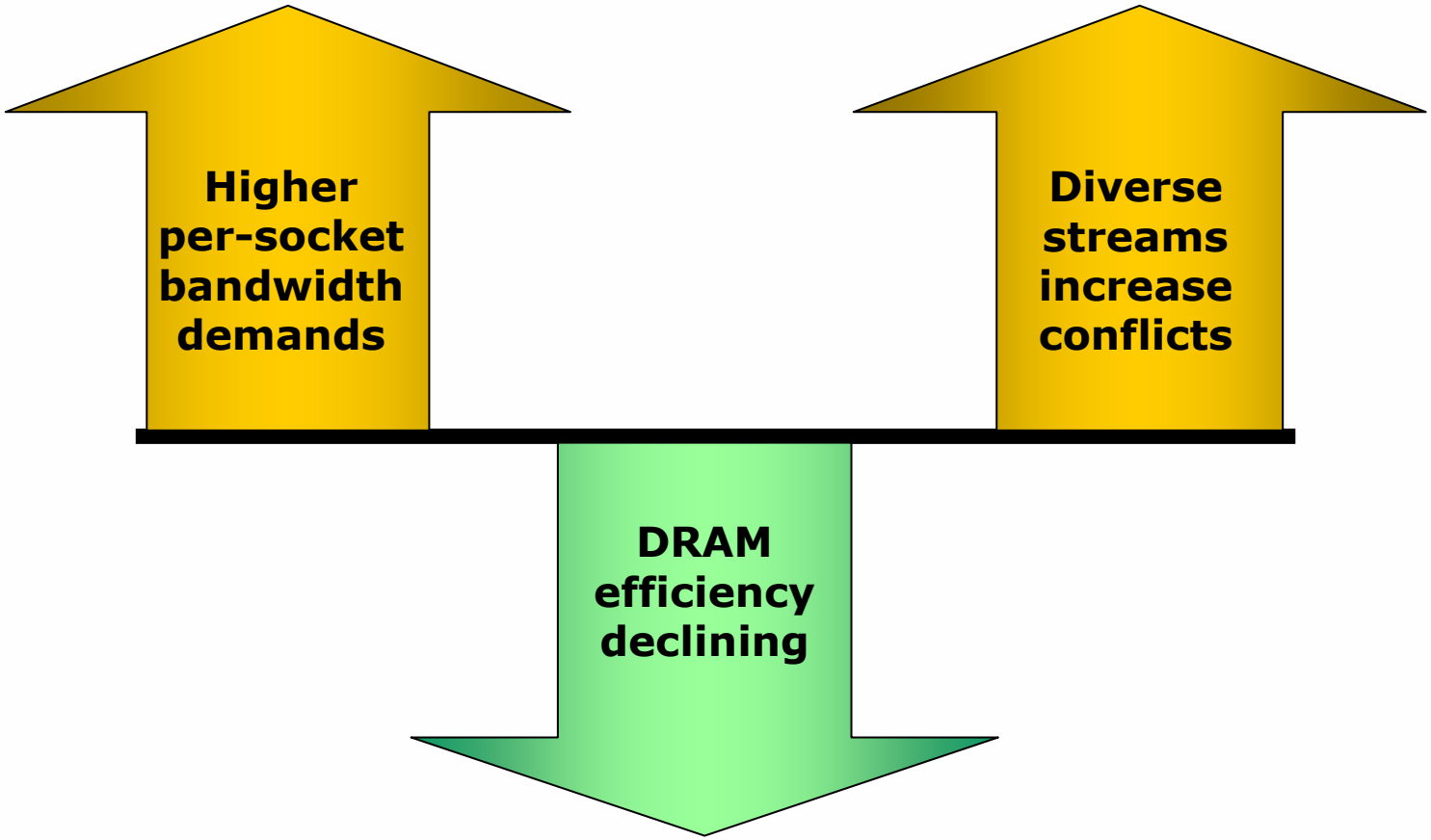
CPU Core IPC Enhancements

AMD
Smarter Choice

- Advanced branch prediction
- 32B instruction fetch
- Sideband Stack Optimizer
- Out-of-order load execution
- TLB Optimisations
- **Data-dependent divide latency**
- **More Fastpath instructions**
 - **CALL and RET-Imm instructions**
 - **Data movement between FP & INT**
- **Bit Manipulation extensions**
 - **LZCNT/POPCNT**
- **SSE extensions**
 - **EXTRQ/INSERTQ,**
 - **MOVNTSD/MOVNTSS**

Trends in DRAM bandwidth

Improved Efficiency is the Answer



We must improve ***delivered*** DRAM bandwidth

Delivering more DRAM bandwidth

- **Independent DRAM controllers**
- Optimized DRAM paging
- Re-architect NB for higher BW
- Write bursting
- DRAM prefetcher
- Core prefetchers

- **Concurrency**
- **More DRAM banks**
 - **reduces page conflicts**
- **Longer burst length**
 - **improves command efficiency**

Delivering more DRAM bandwidth

- Independent DRAM controllers
- **Optimized DRAM paging**
 - **Increase page hits, decrease page conflicts**
 - **History-based pattern predictor**
- Re-architect NB for higher BW
- Write bursting
- DRAM prefetcher
- Core prefetchers

Delivering more DRAM bandwidth

- Independent DRAM controllers
- Optimized DRAM paging
- **Re-architect NB for higher bw**
 - **Increase buffer sizes**
 - **Optimize schedulers**
 - **Ready to support future DRAM technologies**
- Write bursting
- DRAM prefetcher
- Core prefetchers

Delivering more DRAM bandwidth

- Independent DRAM controllers
- Optimized DRAM paging
- Re-architect NB for higher BW
- **Write bursting**
 - **Minimize Rd/Wr Turnaround**
- DRAM prefetcher
- Core prefetchers

Delivering more DRAM bandwidth

- Independent DRAM controllers
- Optimized DRAM paging
- Re-architect NB for higher BW
- Write bursting
- **DRAM prefetcher**
 - **Track positive and negative, unit and non-unit strides**
 - **Dedicated buffer for prefetched data**
 - **Aggressively fill idle DRAM cycles**
- Core prefetchers

Delivering more DRAM bandwidth

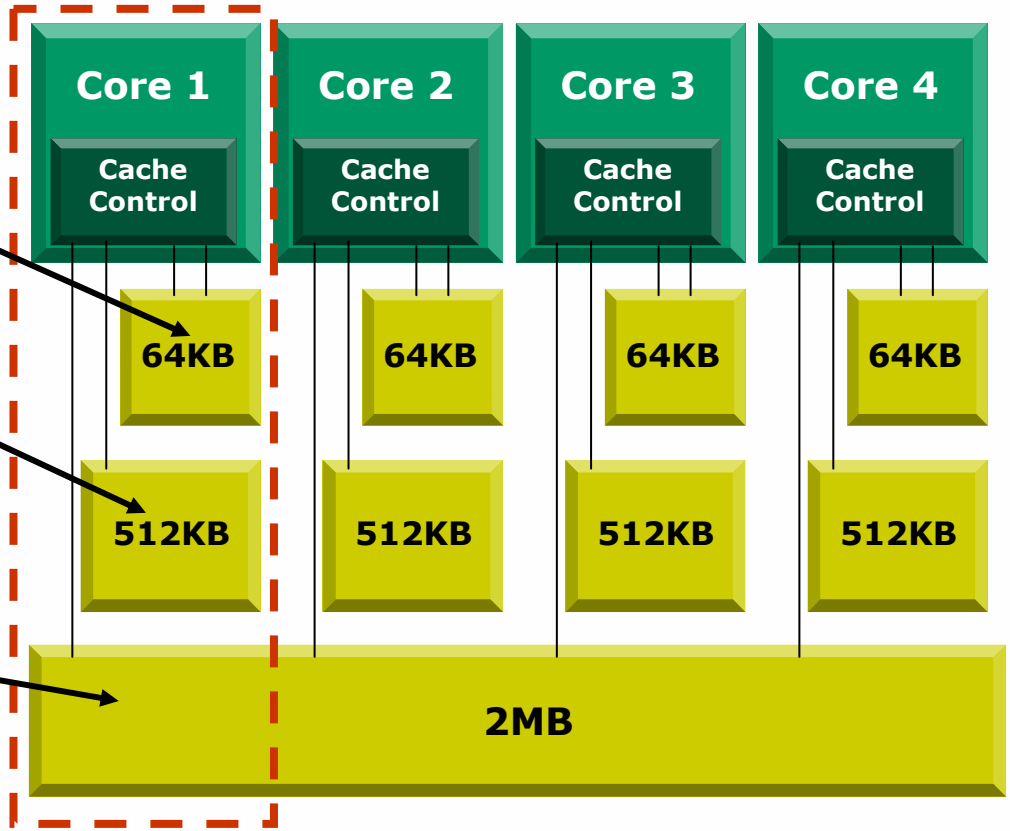
- Independent DRAM controllers
- Optimized DRAM paging
- Re-architect NB for higher BW
- Write bursting
- DRAM prefetcher
- **Core prefetchers**
 - **DC Prefetcher fills directly to L1 Cache**
 - **IC Prefetcher more flexible**
 - 2 outstanding requests to any address

Balanced, Highly Efficient Cache Structure

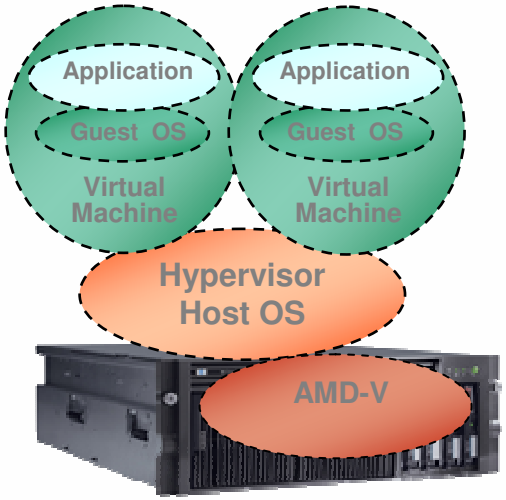
Dedicated L1
- Locality keeps most critical data in the L1 cache
- Lowest latency
- 2 loads per cycle

Dedicated L2
- Sized to accommodate the majority of working sets today
- Dedicated to eliminate conflicts common in shared caches
 - Better for Virtualization

Shared L3 – NEW
- Victim-cache architecture maximizes efficiency of cache hierarchy
- Fills from L3 leave likely shared lines in the L3
- Sharing-aware replacement policy
- Ready for expansion at the right time for customers



Virtualization Background



- Why virtualize?
 - Huge cost savings through consolidation
 - Disaster recovery – can move OSs to a new server.
 - Expect 40% of x86 servers will be virtualized in next 3 years!
 - Consumer applications as well
- AMD Virtualization™ (AMD-V) status
 - Hardware virtualization support available now
 - Simplifies software investment for Hypervisor development
 - Improved security via AMD-V Device Exclusion Vector
 - Better performance from AMD-V tagged TLBs

Virtualized Address Translation

AMD Smarter Choice

- Virtualization adds a new level of address translation in the Hypervisor
 - Guest physical to Host physical
- Current Technology: Shadow Paging
 - Software-only approach to virtualized address translation
 - **Complex and slow**

Faster Virtualization Performance

AMD Smarter Choice

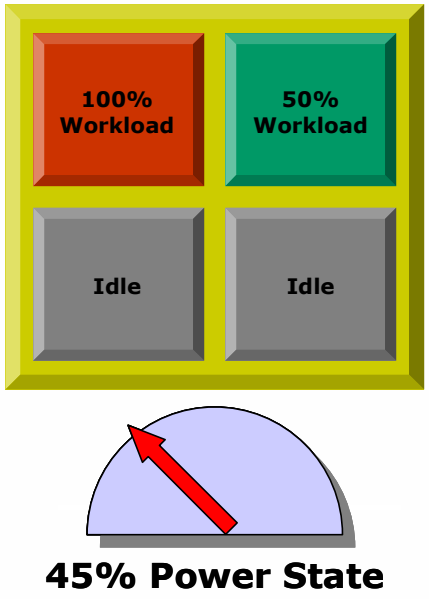
- Nested Paging (NP)
 - Guest and Host page tables both exist in memory
 The processor walks both guest and host page tables
 - Nested walk can have up to 24 memory accesses!
 Hardware caching accelerates the walk
 - "Wire-to-wire" translations are cached in TLBs
 - NP eliminates Hypervisor cycles spent managing shadow pages
 As much as 75% of Hypervisor time

- Barcelona also reduces world-switch time by 25%
 - World-switch time : round-trip to the Hypervisor and back

Advanced Power Management

- **Separate CPU core and Northbridge power planes**
 - **Allow processors to reduce voltage while NB continues to run**
 Power savings
 - **Also can apply additional voltage to NB to raise the NB frequency**
 Performance boost in power-constrained platforms

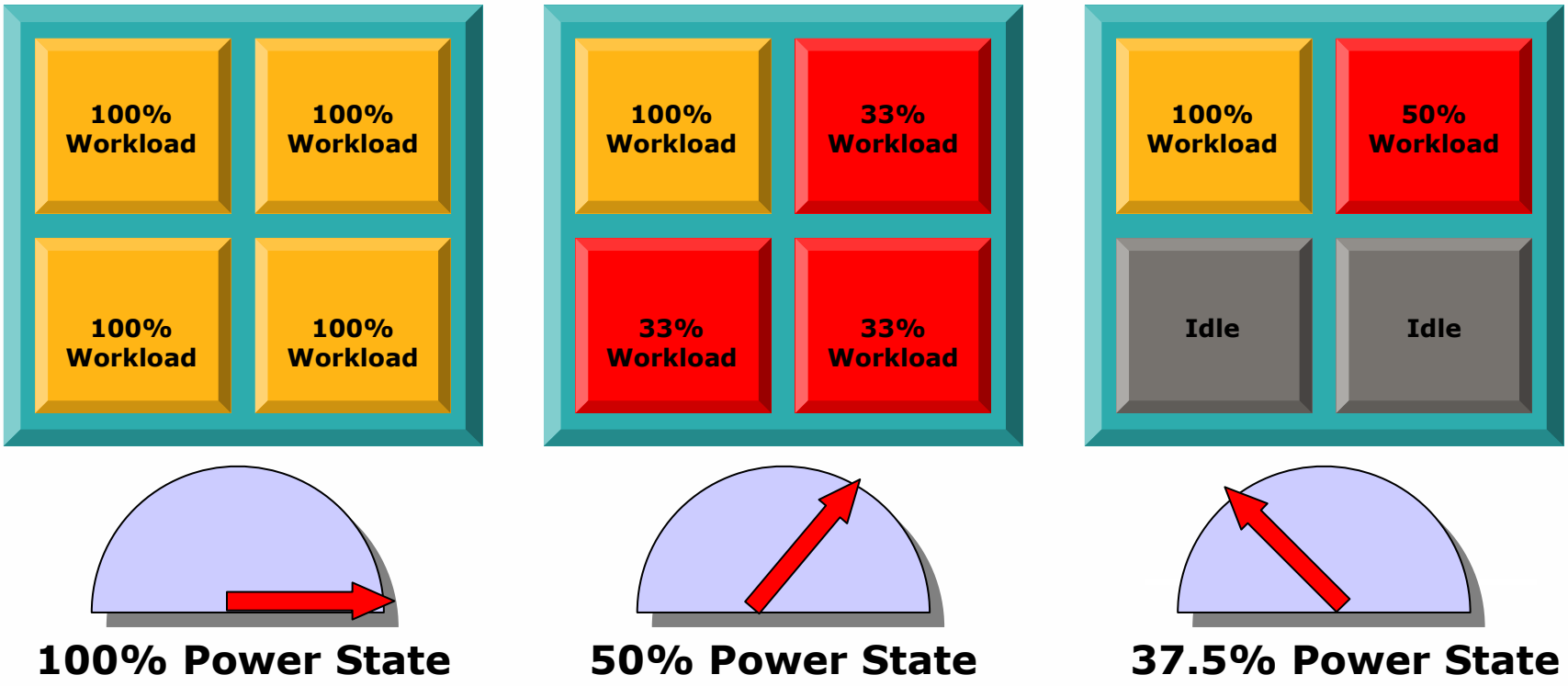
- **Enhanced AMD PowerNow! technology**
 - **Ability to *dynamically* and *individually* adjust core frequencies for improved power efficiency**



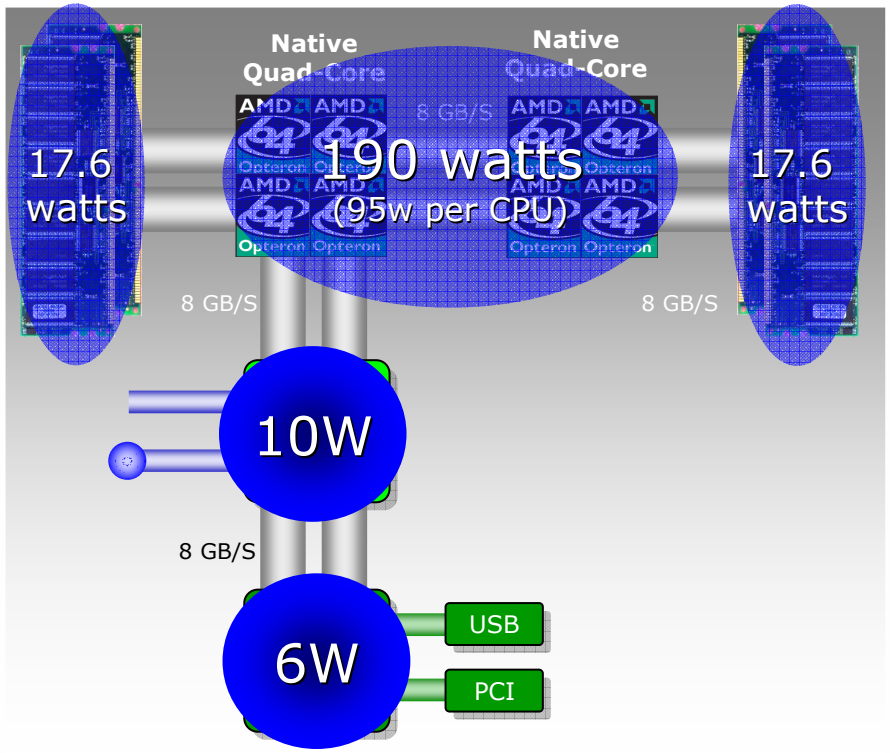
Quad-Core AMD Opteron™ Processor
Enhanced AMD PowerNow!™ Technology with Optimized Power Management

AMD Smarter Choice

Ability to dynamically and individually adjust core frequencies for improved power efficiency



Quad-core System Power



2P/8C System

- 190 watts for processors
- 16 watts for chipset
- 35.2 watts for DDR2

- 241.2 watts total

- Direct Connect Architecture Savings:
- *No external memory controller – saves 25 watts*
- *No FBDIMM – saves 48 watts*

- *System power is the metric that matters to our customers.*
- *Direct Connect Architecture helps reduce system power.*

Summary

- "Barcelona":
 - Native quad-core upgrade planned for 2007
- Processor Details
 - Comprehensive upgrades for SSE128
 Upgrade Execution, Instruction, and Data Bandwidth
 - IPC-enhanced core
 Boost core performance
 - More *delivered* DRAM bandwidth
 Even from the same DRAM technology
 - L3 Cache Architecture
 Shared and expandable
 - Virtualization performance
 Nested paging and faster world-switches
 - Advanced power management
 Optimize system power



Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice



AMD
Smarter Choice

Understanding Multi-Threading

A brief overview of how Multi-threading works

Justin Boggs
Sr. Developer Relations Engineer

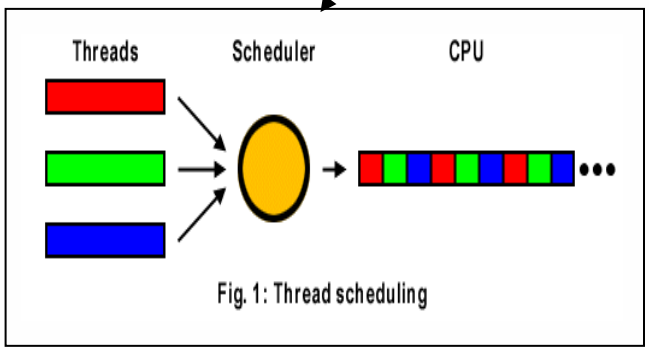
Overview

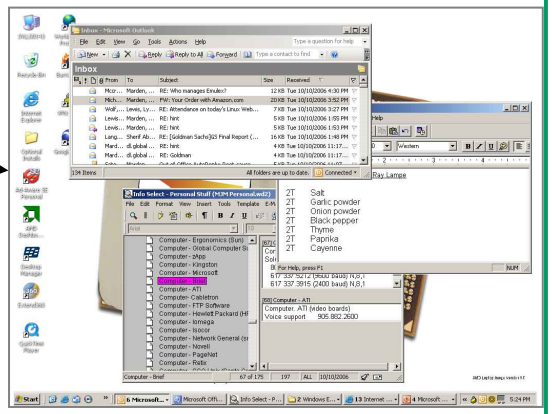
AMD
Smarter Choice

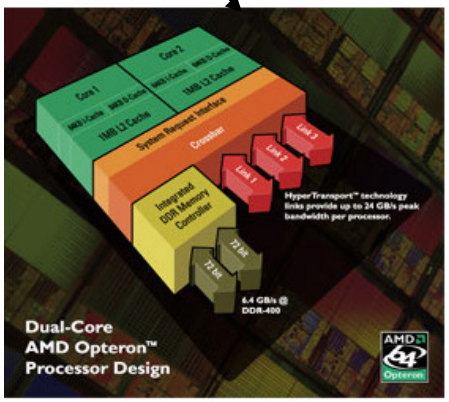
- Multi 'Processor', 'Core', and 'Threading' Comparison
- Multi-Tasking Overview
- Types of Multi-Threading
 - Coarse-Grained Threading
 - Fine-Grained Threading
- Fine-Grained Multi-Threading Options
- Threading Models
- Pitfalls – Safety, Race, Deadlocks, Starvation, Thundering Herds, OS Considerations
- Thread Creation and Synchronization APIs – C, C++, Java, C#
- Considerations for development and OS

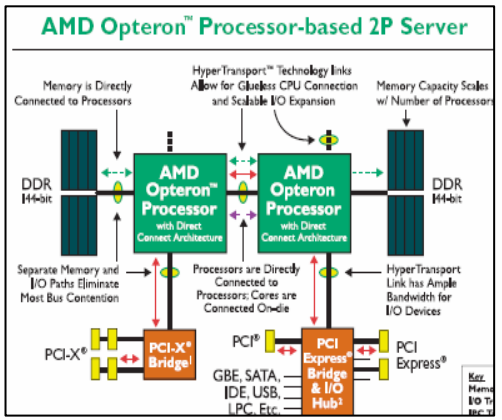
The World now thinks of computer power in Multiples

- Single Tasking
- Multiple Tasks
- Multiple Processors
- Multiple Cores
- Multiple Threads





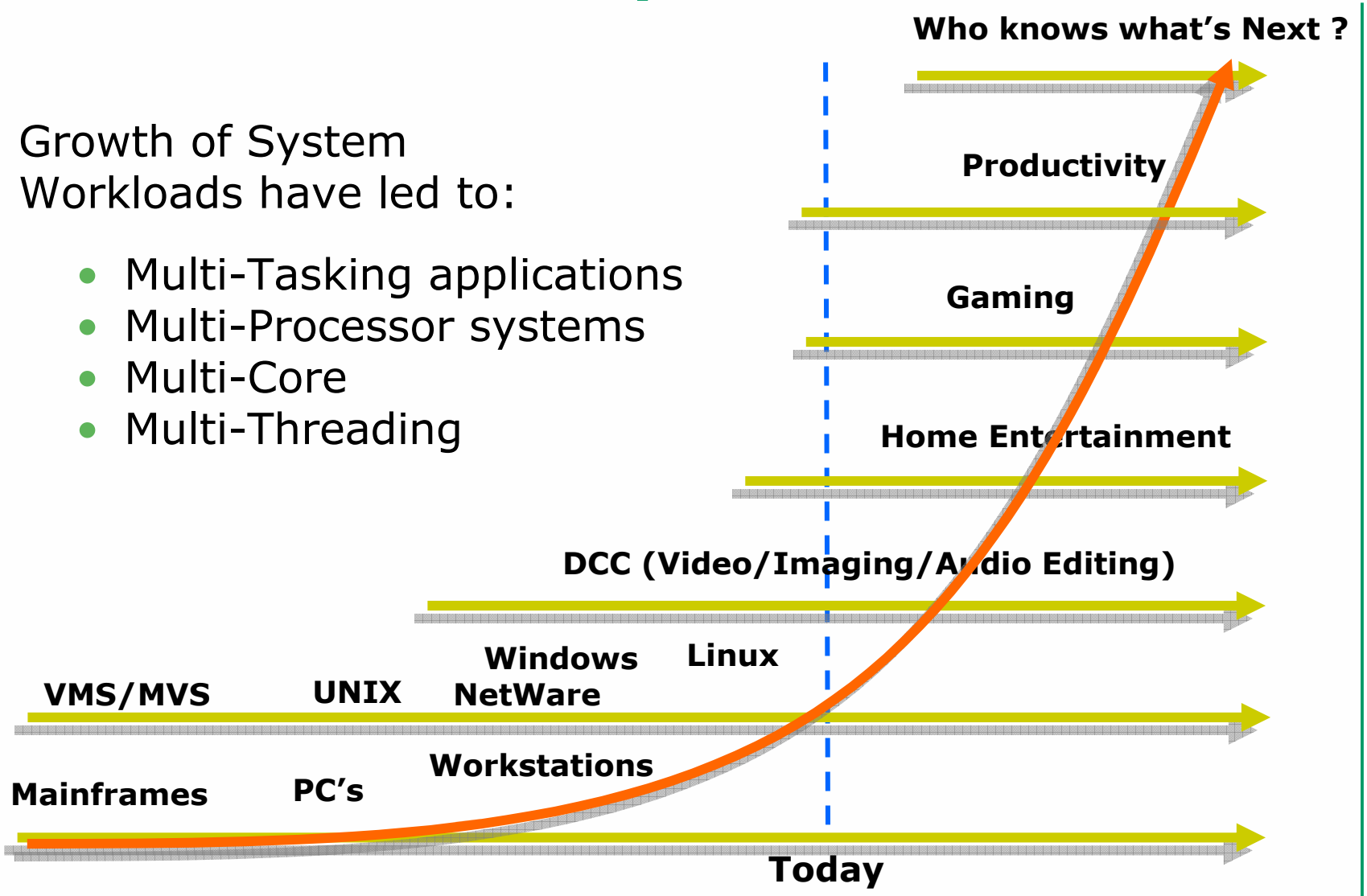




Where are we at today?

Growth of System Workloads have led to:

- Multi-Tasking applications
- Multi-Processor systems
- Multi-Core
- Multi-Threading



What is Multi-Tasking?

AMD
Smarter Choice

- In General:
 - A task is… well just "doing something"
 - Multi-tasking doing more than one 'thing' at a time… or really *switching* between things…

Multi-tasking right here in this room

- Watching a presentation about Multi-threading
- Thinking about your Security Audit next week
- Checking an email
- Deciding what to have for dinner tonight

Multi-tasking in your car

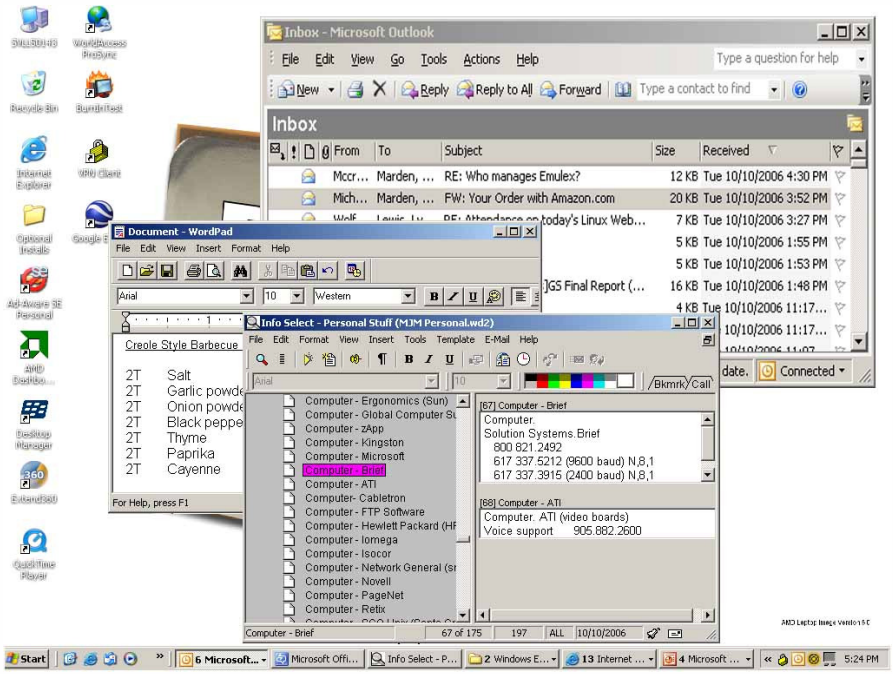
- Steering
- Maintaining your speed
- Observing traffic
- Listening to the radio

What is Multi-Tasking?

- In a Computer environment:
 - A task is an app or process that uses:
 - *One or more Code, Data, and Stack segments*
 - *It's own Local Descriptor Table*
 - *It's own Page Translation Table*
 - Multi-tasking Running disparate, non-related applications
 - Not sharing memory or data

<u>Multi-tasking on the computer</u>

- Email
- Word-processing
- Spreadsheet
- Recording some data to CD



What are Threads?

AMD
Smarter Choice

- Unit of work scheduled on a processor
- Threads 'appear' to run in parallel (concurrent)
- Making your apps faster by utilizing the OS, processor, and memory
- Decomposing one larger process into two or more smaller processes
- Is Multi-tasking the same as Multi-threading ?
 - Could be considered 'Coarse' Multi-Threading <- time-slicing
- The "free lunch" is over with respect to significant increases in single-threaded execution
 - …but I want it to perform out of the box unchanged
 - I want to have a 10GHz machine
 - Single-core performance not increasing as in the past
 - Dual-core processors are proliferating everywhere
 - Without multi-threading, cores will be underutilized, wasting otherwise usable CPU cycles

Threads vs. Processes

AMD Smarter Choice

- What they have in common
 - Have an id, priority, and scheduling policy
 - Function independent from the parent process
 - Share resources with the parent process
 - Can create resources
 - The creator can exercise control over the thread or process
 - Can not access the resources of another process

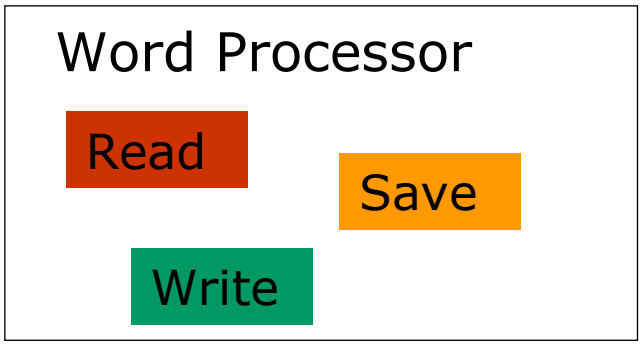
- Advantages of Threads
 - Share the same address space
 - Have very little overhead
 - Can communicate with other threads directly

- Disadvantages of Threads
 - Requires synchronization to write memory (race conditions)
 - Address space corruption
 - Timing conditions

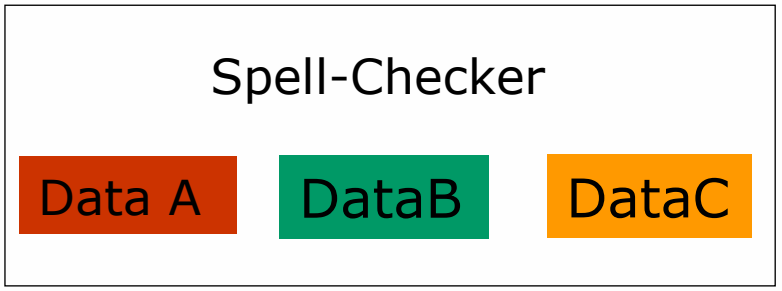
Different methods of Threading

- Coarse-grained Multi-threading (functional)
 Splitting up the different components/tasks within one process
- Fine-grained Multi-threading (data-parallel)
 Breaking down one task by processing different portions of data using the same method for each data set

Coarse-grained

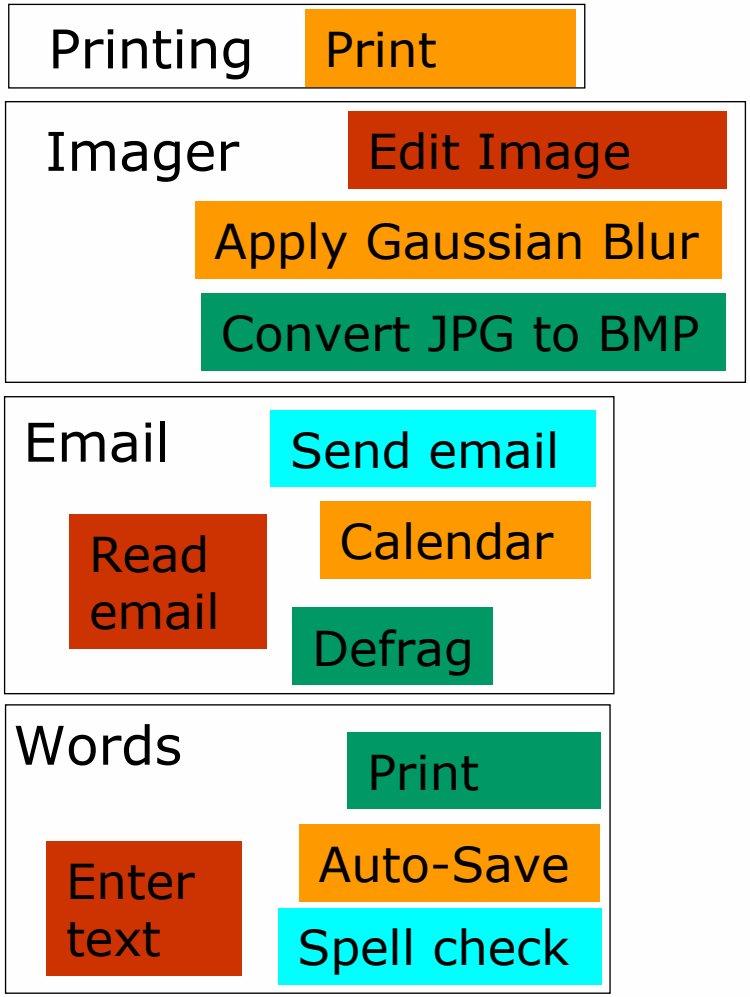


Fine-grained



Coarse-grained Threading

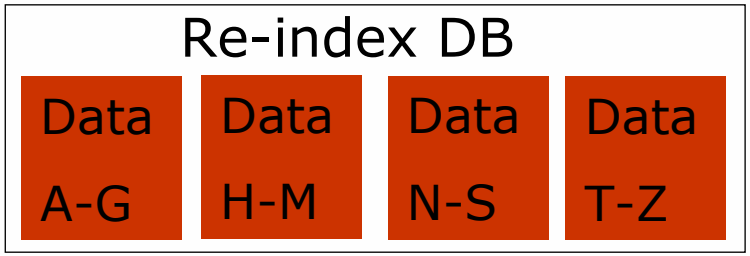
- Splits up different components within one process
- Examples:
- Printing
- Image editor
 - Open up image for editing
 - Apply Gaussian Blur
 - Convert image to different format
- Email
 - Read message
 - Sending and receiving email
 - Processing Calendar alerts
 - Defragment message store
- Word Processor
 - Enter text
 - Spell-checker
 - Print document
 - Auto-Save document to disk

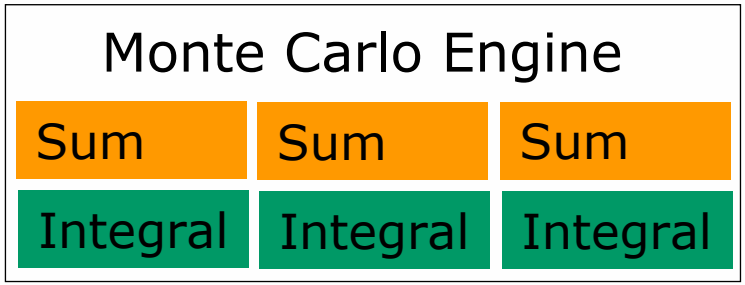


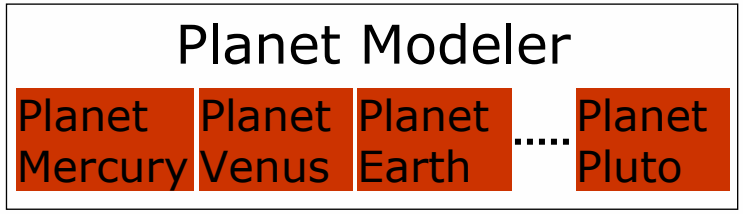
Fine-grained Threading

- Break up data or events in application and process across multiple execution threads at the same time to complete the task faster
- Inherent parallel applications – Lots of repetitive processes

- Database Server
 - Process multiple queries
 - Re-index the database
 - Multiple databases open
- Monte Carlo Engine
 - Calculate values for historical data by breaking data down and processing across multiple execution threads
- Modeling Planetary Motion
 - Calculate the orbit of all the individual planets with solar system







Methods of Fine-grained Threading

AMD
Smarter Choice

Implicit

- Relying on your Compiler and API Libraries to do the work for you

- Explicit

- Doing it yourself
- Lots of work
- Best handled by experts

Why do Implicit Threading?

AMD Smarter Choice

- Let the compiler and libraries do all the work
- Although you may not achieve 'optimal' performance, the performance may be "much better"
- It's easier.
 - You don't have to do much.
 - However…you also don't have as much control
- However, it may not be consistent
 - What if your code-base changes ?
 - What if your compiler changes ?
 - Less control. Compiles to a least common denominator
- Who should do this ?
 - You don't have the time
 - You don't have the training
 - You don't have the expertise...

Implicit Threading

Some examples of Compiler Optimisations

AMD
Smarter Choice

- Compiler <– *looks at dependencies*
- Induction variable elimination
- Strength reduction
- Nested Loop vectorization
- Nested Loop parallelization
- Loop fusion
- Loop invariance removal
- Loop invariant conditional hoisting
- Loop interchange
- Loop splitting/peeling
- Loop Unrolling and jamming
- Loop Un-switching
- Software pre-fetching
- Scalar expansion & replacement

Why do Explicit Threading ?

The next level down

AMD
Smarter Choice

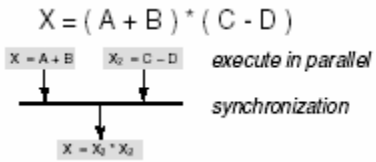
- The compiler and libraries may not be able to Multi-thread your application bottlenecks
- You can achieve the best possible performance !
- It's a lot of work. You may run into problems. Race conditions, livelocks, deadlocks, resource starvation, OS considerations
- Maintain better consistency. Less dependent on compiler changes ?
- You can control how tasks are multi-threaded
- Who should do this ? Experts only !!!

Next, choose the Threading Model ?

Data-Level or Instruction-Level Parallelism

AMD Smarter Choice

- Data-level Parallelism (DLP)
- Multiple threads which do the same work on different sets of data
- Compute-intensive loops - the same operation is applied repeatedly to different data
- Can scale to the number of cores on the machine



Note: When scheduling multiple threads which access independent data on an idle system, it is preferable first to schedule the threads to an idle core of each node until all nodes are exhausted and then schedule the other idle core of each node. In other words, schedule using node major order first, followed by core major order.

- Examples:
 - Processing appointments in a calendar
 - Processing images or portions of images
 - Word Processing – editing multiple documents at the same time
 - Spell-checker

Next, choose the Threading Model ?

Data-Level or Instruction-Level Parallelism

AMD
Smarter Choice

- Task/Instruction-level Parallelism (ILP)
- Running different tasks (multiple events)
- Superscalar hardware picks up most available ILP (complexity effective)
- Encapsulating 'Independent' functions of work
- Thread libraries (e.g., the Win32* thread API or POSIX* threads) are designed to express task-level concurrency

- Examples:
 - Computer game - threads for audio, graphics, and strategy
 - Email – composing, receiving, archiving, and reading email messages
 - Word Processing – entering text, spell-checking, auto-saving document
 - Web server – Serving multiple user requests

Creating Threads

AMD Smarter Choice

- **Thread Creation Libraries:**
- C++

 Win32

 createthread, endthread

 POSIX

 Supports parallel processing within a single application at the function or object level. Can be used to take advantage of SMP or MPP.

 OpenMP

 MPICH (MPICH2)

 Supports large-scale, complex cluster programming. Strong support for SPMD model. Also supports SMP, MPP, and multiuser configurations.

 PVM

 Supports cluster programming of heterogeneous environments. Easy to use for single-user, small to medium cluster applications. Also supports MPP.

 MICO

 Supports either distributed or object-oriented parallel programming. Contains nice support for agent and multiagent programming.
- Java

 Create a Class with the Runable interface and then define a 'run' Method
- Delphi

 Derive a class from TThread, and implement its Execute() method. You can create a unit with a template thread class in it by selecting New/Thread Object from the File menu

Multi-threading: Pitfalls

AMD Smarter Choice

- Thread Safety — Reentrancy issues (static and global data accesses, use mutexes)
-
- Deadlocks — Proverbial chicken and egg
- Livelocks — Not blocked, never finishes
- Thread Starvation — Priority execution issues
- Scheduler Thrashing — Thundering Herds from single event
- Race conditions — Unpredictable, concurrency problems
- System Considerations — Timestamp counter
 Windows XP SP2 hot fix for Timer
 NUMA support
 Determine number of Cores
 Windows: GetLogicalProcessorInformation()

Multi-threading: Pitfalls

AMD
Smarter Choice

Do not use the "RDTSC" timestamp counter!

- "RDTSC" timestamp instruction works just the same as always…
 - … but each core has its own timestamp counter in a multi-core system…
 - … and they are *not* guaranteed to be in sync…
 - … therefore **comparing two values from RDTSC can be dangerous.**
- Windows will move your process between cores
- Use:
 - `QueryPerformanceCounter()`
 - `QueryPerformanceFrequency()`
- See "Game Timing and Multicore Processors" at MSDN: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/Game_Timing_and_Multicore_Processors.asp

Multi-threading: Suggestions

AMD
Smarter Choice

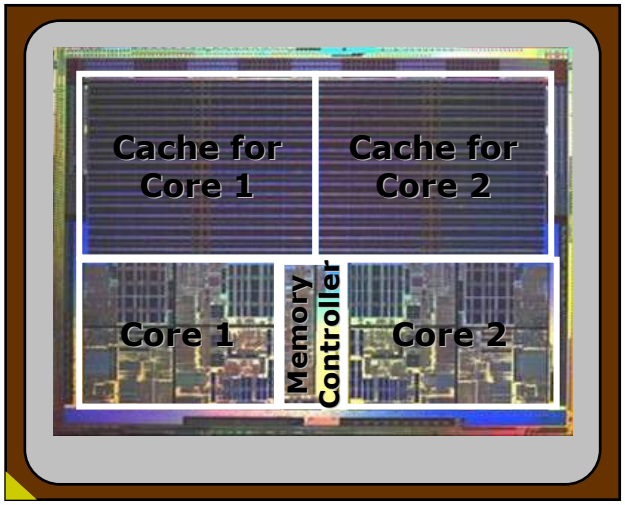
- General rules for good multi-threading
 - Minimize data sharing between threads
 - Synchronize rarely
 - Good use of semaphores
 - Hold locks briefly

- How many threads
 - Tune the number of threads you create based on number of processors in the system

- Generally, one thread per core is ideal
 - Except if your threads are not very busy and tend to block
 - Too many threads will cause excessive context switches, and hurt performance, so measure

Multi-threading: Benchmarking tips

- Use this option in your boot.ini file /numproc=1
 - Windows XP will only use 1 of the cores
- Process affinity
 - Set process affinity in Task Manager to only one processor
- Easy to benchmark single-core vs. dual-core performance
- All other system components & config. remain constant



Cache Can Largely Impact Performance

AMD Smarter Choice

- Maximize benefit from the cache in dual-core and beyond
- Each AMD Athlon™ 64 processor or AMD Opteron™ Processor core has its own separate L1 caches
- ***Each core also has its own separate L2 cache***
- Greater parallelism enables added performance:
 - L2 cache latency does not increase when another L2 is added
 Simply increasing the size of a shared cache would increase latency
 - Total cache bandwidth scales up linearly with the number of cores
 - Threads cannot evict another thread's data from other L2
- L2-per-core gives *symmetric* scaling beyond dual-core

Cache Impact on Threading Architecture

AMD Smarter Choice

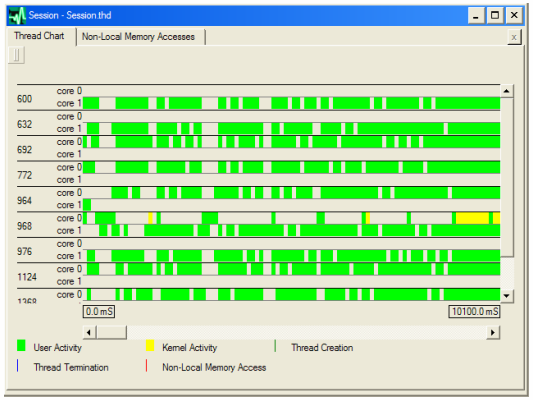
- Avoid "cache thrashing" between cores
 - Avoid multiple threads writing to the same variables (of course)
 - Also avoid one thread frequently writing what another is reading
 - This is just a standard rule of threaded programming
 - Beware of **false sharing** which can cause thrashing:
 - Two threads modifying **different variables** which occupy the **same cache line**
 - Can happen in heap data or in global variables
 - One safe approach: `_aligned_malloc` for 64-byte aligned heap chunks
 - Another handy alignment trick: `__declspec(align(64))`
 - See MSDN for complete details on managing alignment

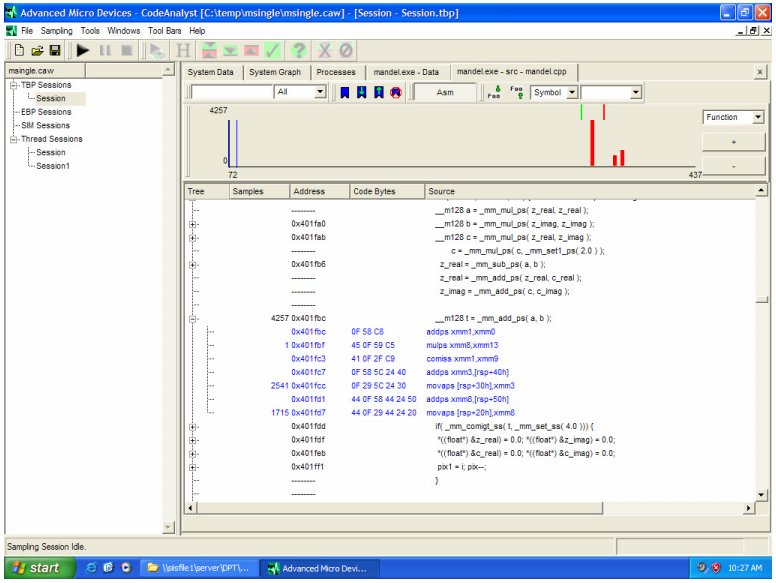
- Use AMD CodeAnalyst™ profiler to examine threads and cache events
 - Thrashing would appear as excessive cache refill events
 - Just one way AMD CodeAnalyst can help you build faster code
 - Available at http://developer.amd.com/

Multi-threading: Use CodeAnalyst

- Identifies threads in the target application
- Shows thread creation and termination
- Monitors CPU affinity of each thread
- Identifies non-local memory access
- Graphs thread activity on each CPU

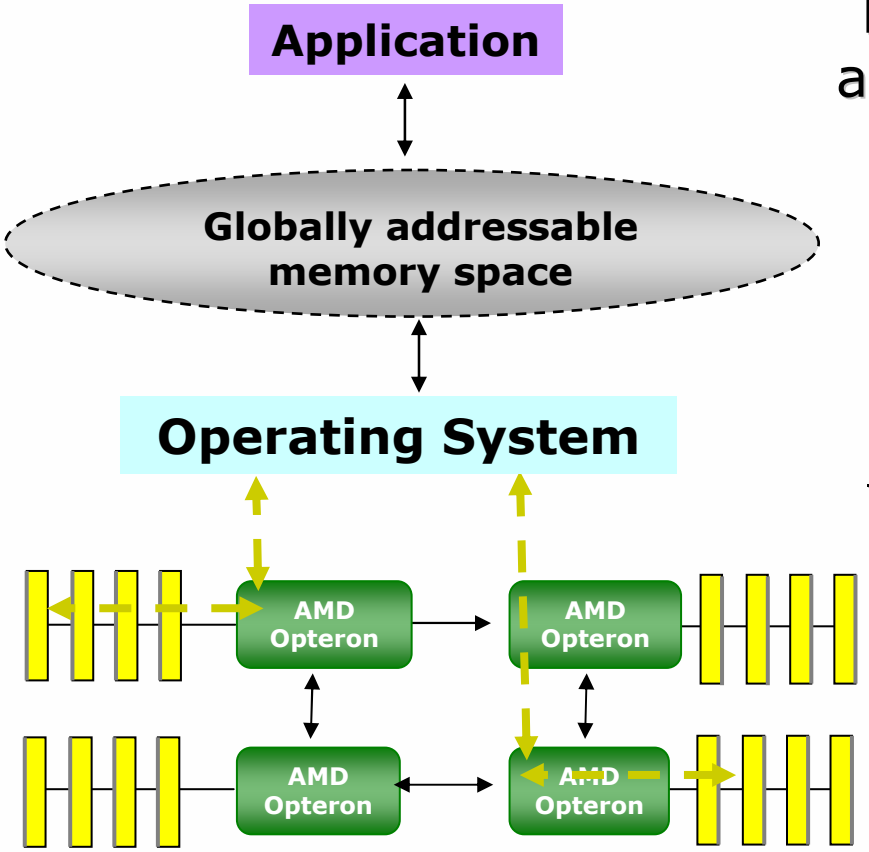
- Timer-based
- Event-based





Thread Profiler shows a thread execution timeline

Non-Uniform Memory Access (NUMA)



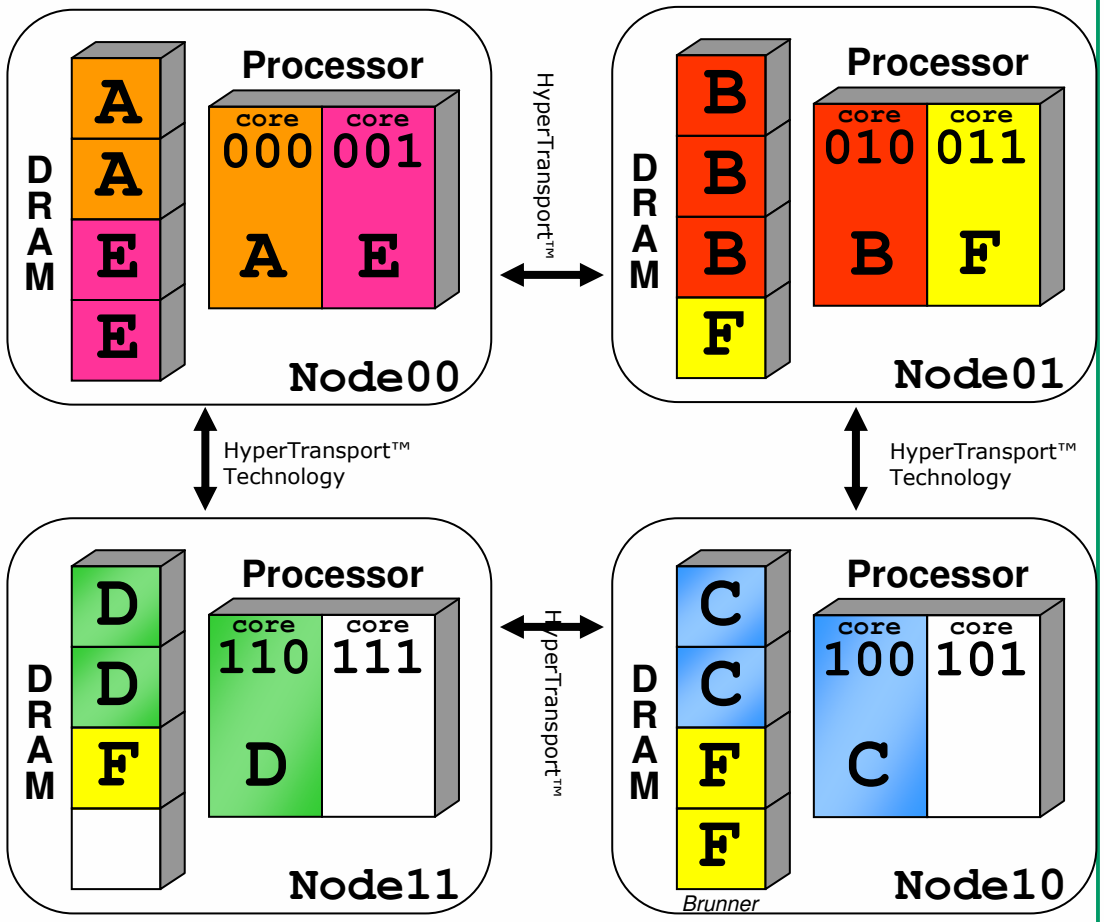
Memory initialized into globally addressable memory space with processors maintaining cache coherency across this space

OS assigns threads from same process to the same NUMA node

Processor has local memory and uses HyperTransport™ for high-speed access to non-local memory

AMD64 Operating System Update
NUMA Scheduling Example

- OS distributes threads across processors
- OS attempts to map a thread's memory requests to local memory
- If no memory is left on the current node, OS will use memory of other nodes



Windows® ccNUMA support

- Microsoft® Windows OS fully supports the AMD Opteron™ NUMA system architecture

- ccNUMA = cache coherent NUMA
- Relevant in multi-socket systems, more than one memory controller

- Windows XP Pro x64 and Server 2003 SP1 support NUMA
- Windows Vista and beyond will have the best NUMA support

- Memory affinity is optimized
 - The OS allocates physically local memory when possible
- ccNUMA APIs can be used for additional control
 - GetNumaNodeProcessorMask, SetThreadAffinityMask, etc.

Windows NUMA APIs

- GetNumaHighestNodeNumber()
 Retrieves highest numbered node in system, but doesn't guarantee that nodes are sequentially numbered.
- GetProcessAffinityMask()
 Retrieves the list of processors on the system
- GetNumaProcessorNode()
 Returns NUMA node for specified processor.
- GetNumaNodeProcessorMask()
 Retrieves list of all processors for a node.
- SetProcessAffinityMask()
 Sets affinity for all threads in process.
- SetThreadAffinityMask()
 Sets affinity for individual thread.
- GetNumaAvailableMemoryNode()
 Retrieve the amount of memory available to a node.

ccNUMA: thread affinity on Windows®

AMD Smarter Choice

- Typically just let Windows manage affinity
 - The OS knows everything
- Sometimes explicit affinity calls will improve performance
 - Be very careful
 - Don't assume anything about the NUMA topology
 - Don't make the affinity more strict than necessary
- NUMA APIs are documented at MSDN
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/numa_support.asp

ccNUMA: memory affinity on Windows®

AMD Smarter Choice

- Windows XP Pro x64 and Server 2003
 - Create thread in suspended state
 - Set thread affinity to a particular NUMA node
 - Start thread
 - Allocate your memory
 - Initialize the memory to cause page faults

 Then the OS physically commits the memory

- Windows Vista/Longhorn is slightly different
 - Create thread in suspended state
 - Set thread affinity to a particular NUMA node
 - Start thread
 - Allocate your memory
 - No need to initialize the memory (but no harm done)

 The OS physically commits the memory on allocate

Detecting the # of Cores Can Be Tricky

AMD Smarter Choice

- Anticipate the adoption of virtual environments
 - Physical computer may be partitioned: you don't get the whole thing
 - Trust the OS to tell you about available resources

 Really try to avoid using low-level instructions on the hardware

- For a recent OS (Vista and later) use the new function **`GetLogicalProcessorInformation()`**
 - Gives you more complete info about NUMA, cache, etc.
 - Lets you distinguish true cores from SMT threads (pseudo-core)

- For Windows XP / 2000 **`GetSystemInfo()`** tells you how many processors you have

Resources for More Information

http://developer.amd.com/resourcecenter_mc.jsp

AMD Smarter Choice

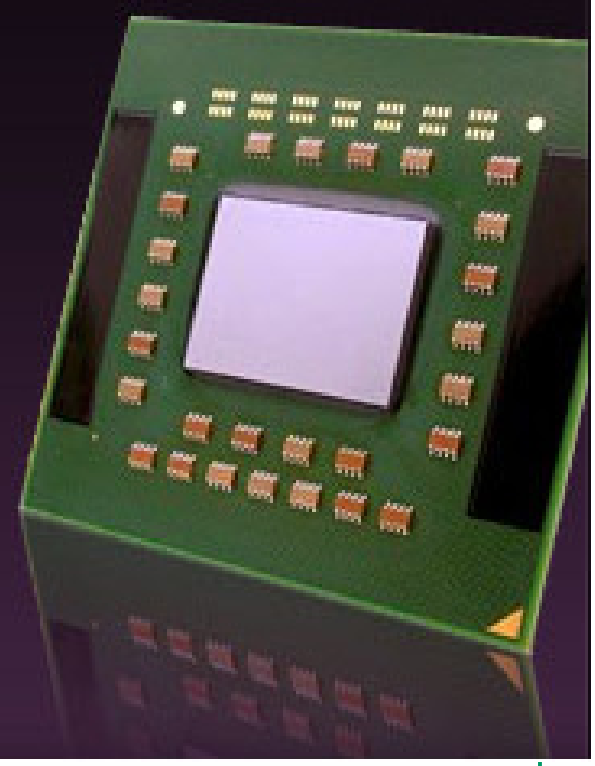
AMD

Home | Login |

DEVELOPER RESOURCES | DEVELOPER COMMUNITY | AMD INFO

WELCOME TO THE
MULTI-CORE RESOURCE CENTER

Multi-core processors represent a major evolution in computing technology. Placing two or more powerful computing cores on a single processor opens up a world of important new possibilities. Learn about the evolution of multi-core technology and how to take advantage of these new opportunities through the use of multi-threading techniques.



Resources for More Information

http://developer.amd.com/resourcecenter_mc.jsp

AMD Smarter Choice

- **Featured Articles**
- Performance of Optimisation of 64-bit Windows Applications for AMD Athlon 64 and AMD Opteron Processors using Microsoft Visual Studio 2005
- Multi-threading for rookies
- What every dev should know about multi-threaded apps
- Microsoft says multi-cores no good without software threading
- MPI

- **Developer Guides**
- AMD64 Architecture Programmer's Manuals:
 - Volume 1: Application Programming
 - Volume 2: System Programming
 - Volume 3: General-Purpose and system Instructions
 - Volume 4: 128-Bit Media Instructions
 - Volume 5: 64-Bit Media and x87 Floating-Point Instructions
- Software Optimisation Guide for AMD Athlon and AMD Opteron
- BIOS and Kernel Developer's Guide for AMD Athlon 64 and AMD Opteron



Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice



AMD
Smarter Choice

Tools & Optimisation

Hervé Chevanne – Dr. Ing.
Applications Analyst

Tools Agenda

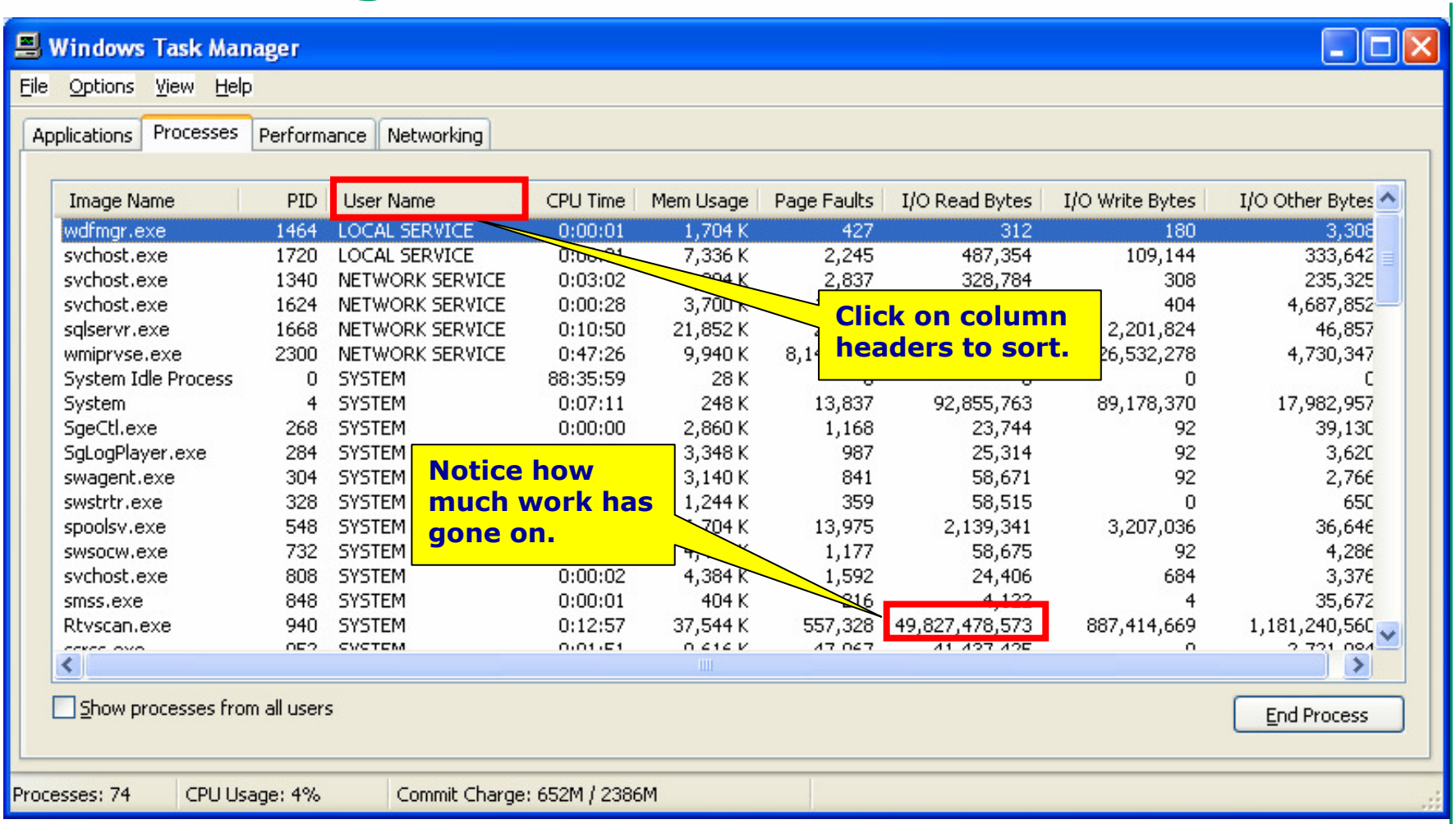
- Window System Profilers
 - Windows Task Manager
 - Performance Monitor (PERFMON)
 - Logman
 - Recommended Counters
- Linux/Unix System Profilers
 - Command-based system profilers
 - Solaris DTrace
- Source Profilers
 - Quantify
 - PGPROF, PGDBG
 - Oprofile
 - CodeAnalyst

Windows Task Manager

AMD
Smarter Choice

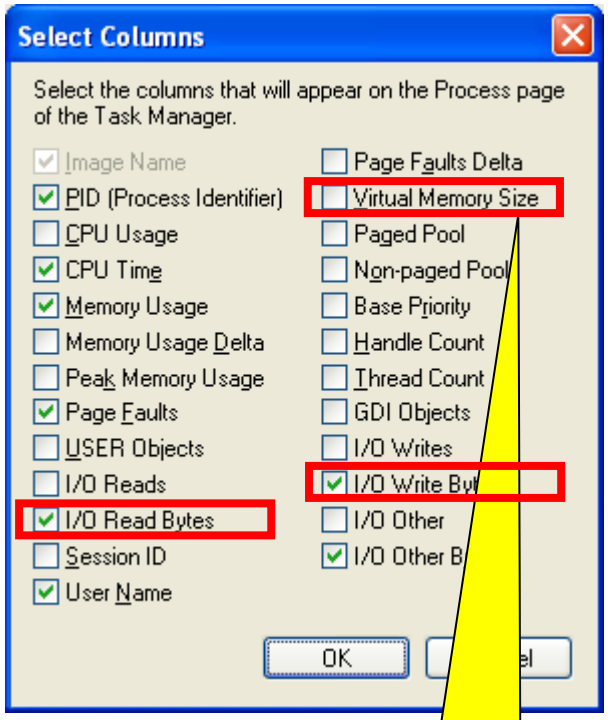
- **Primary tool used on Windows:**
- **Interactive, real-time GUI**
 - Monitor performance of processes, CPU, network
 - Graphical view
 - Almost all values available within Performance Monitor
 - **Kill** applications or processes
 - Change process **priority**, set CPU **affinity**
- **Applications Tab**
 - Lists programs represented with a window
 - Start new programs
- **Processes Tab**
 - Program or process name
 - Image Name, PID, User Name, Session ID columns
 - I/O, Memory columns
 - GDI Objects, Handle Count columns
 - Values displayed are totals
- **Performance Tab**
 - Charts CPU utilization
 - Some memory utilization totals
- **Networking Tab**
 - Charts your network adapters

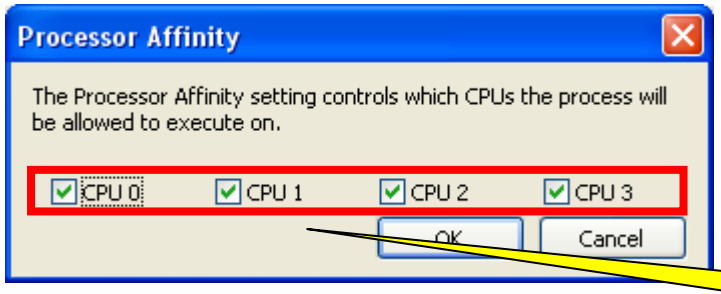
Task Manager – Processes Tab

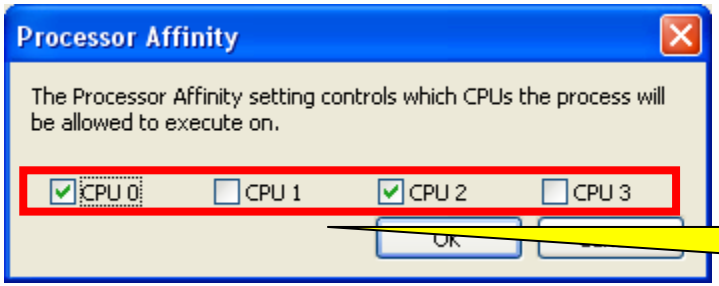


Task Manager – Processes Tab

- View -> Select Columns
- The I/O Read Bytes and I/O Write Bytes are very useful running totals for each process.
- Set Processor Affinity





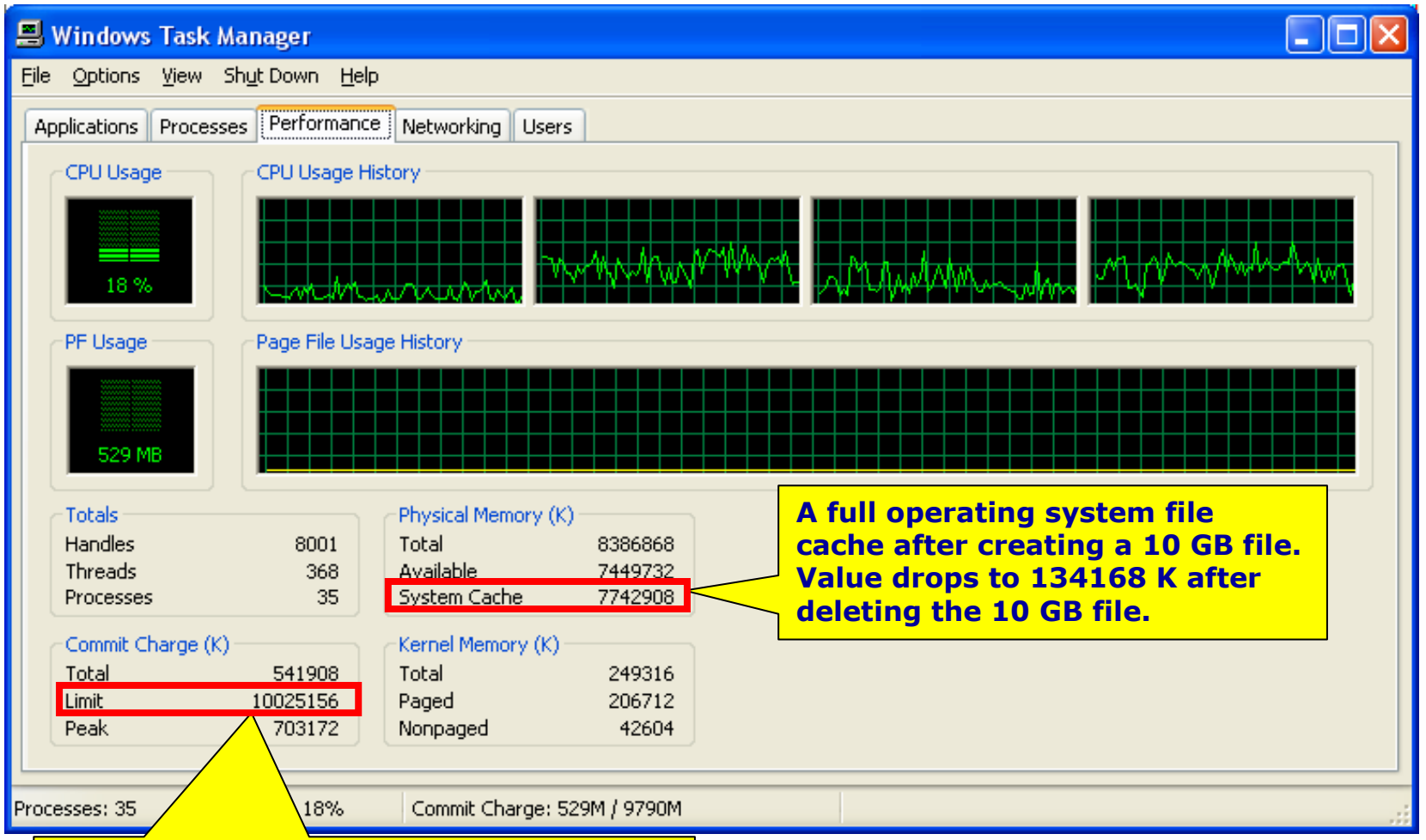


Affinity to all four cores.

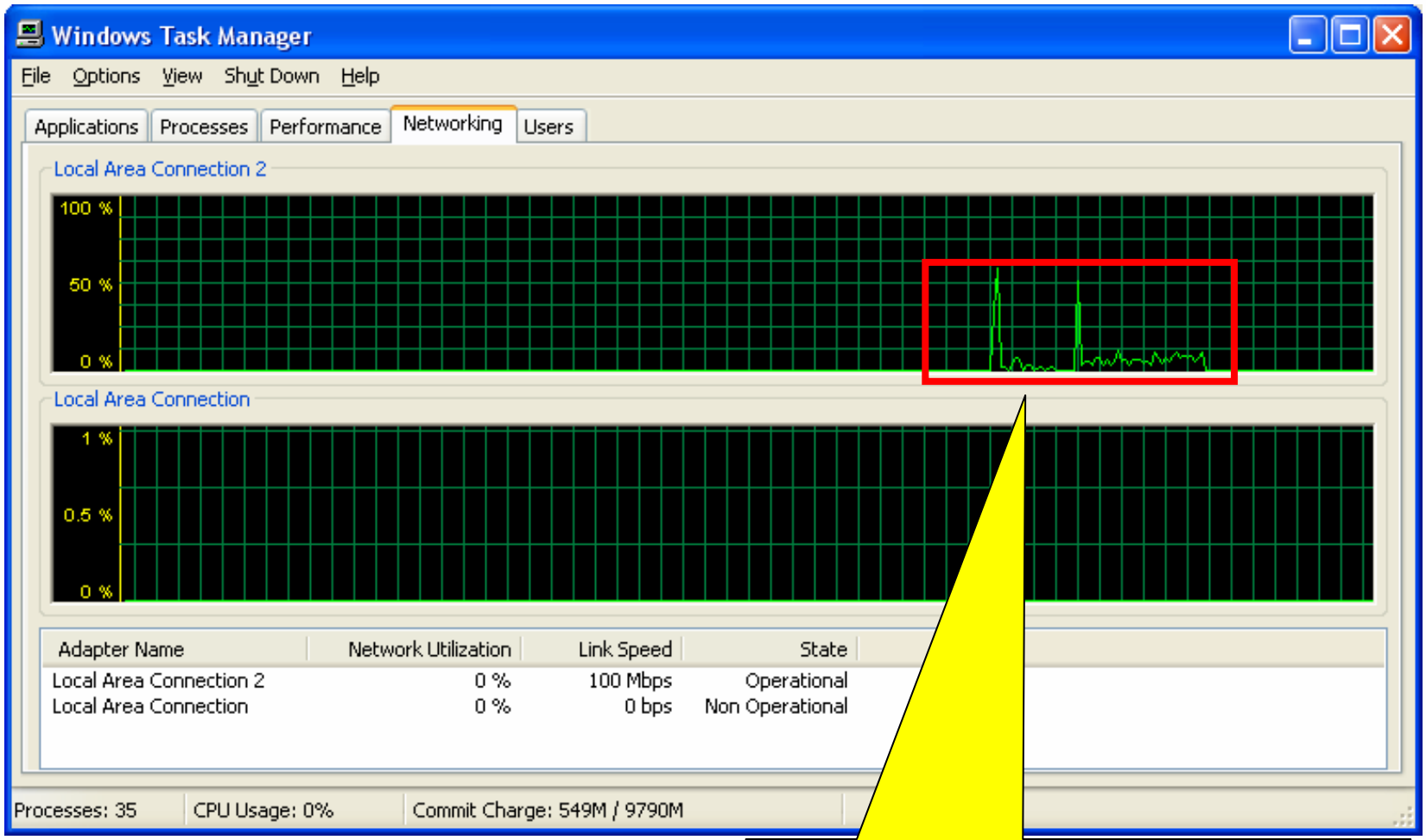
Virtual Memory Size also very useful.

Affinity to first core on first processor and first core on second processor.

Task Manager – Performance Tab



Task Manager – Networking Tab



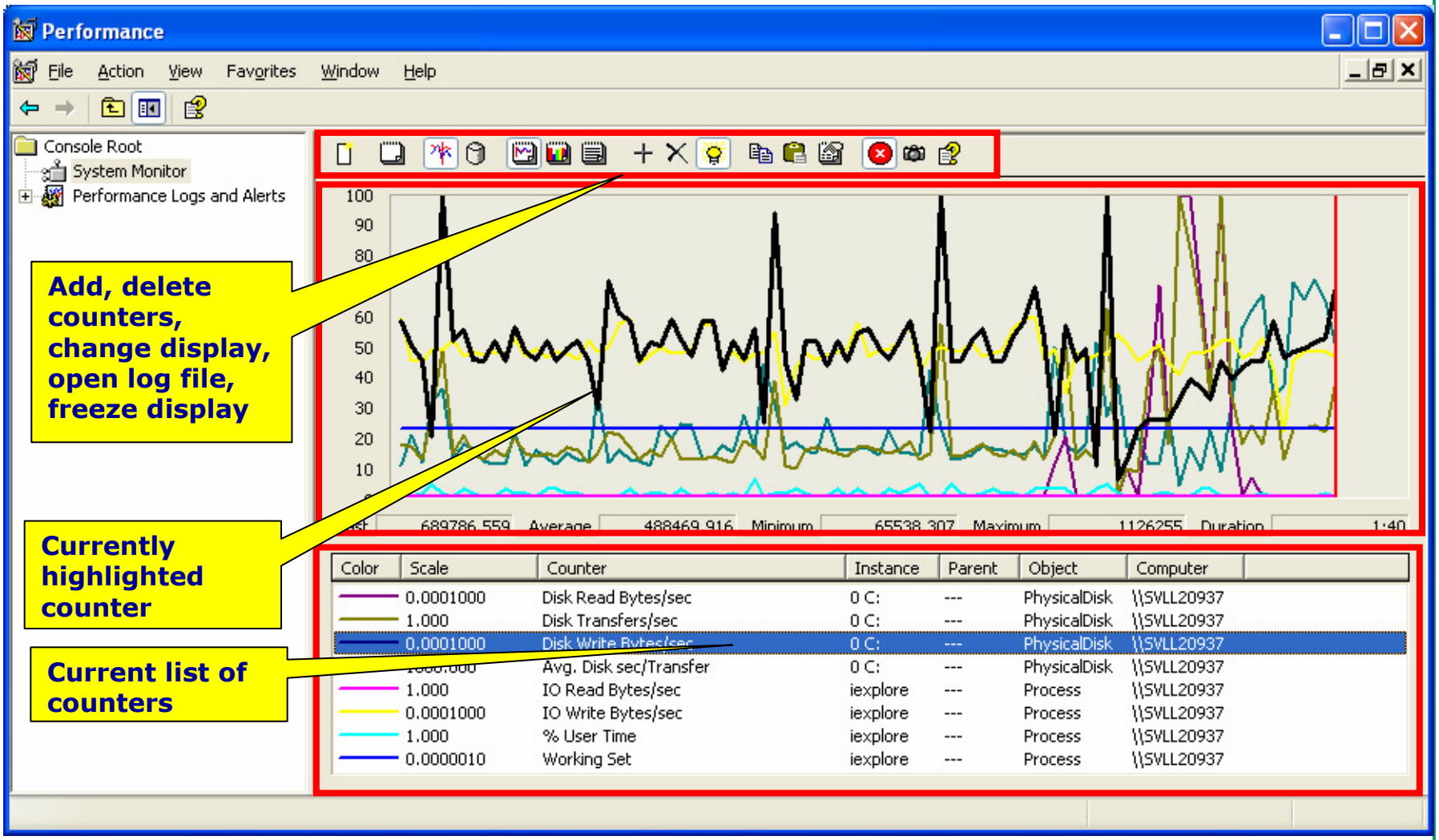
Spike in traffic from a 50MB ftp at about 800 K/sec average.

Windows Performance Monitor

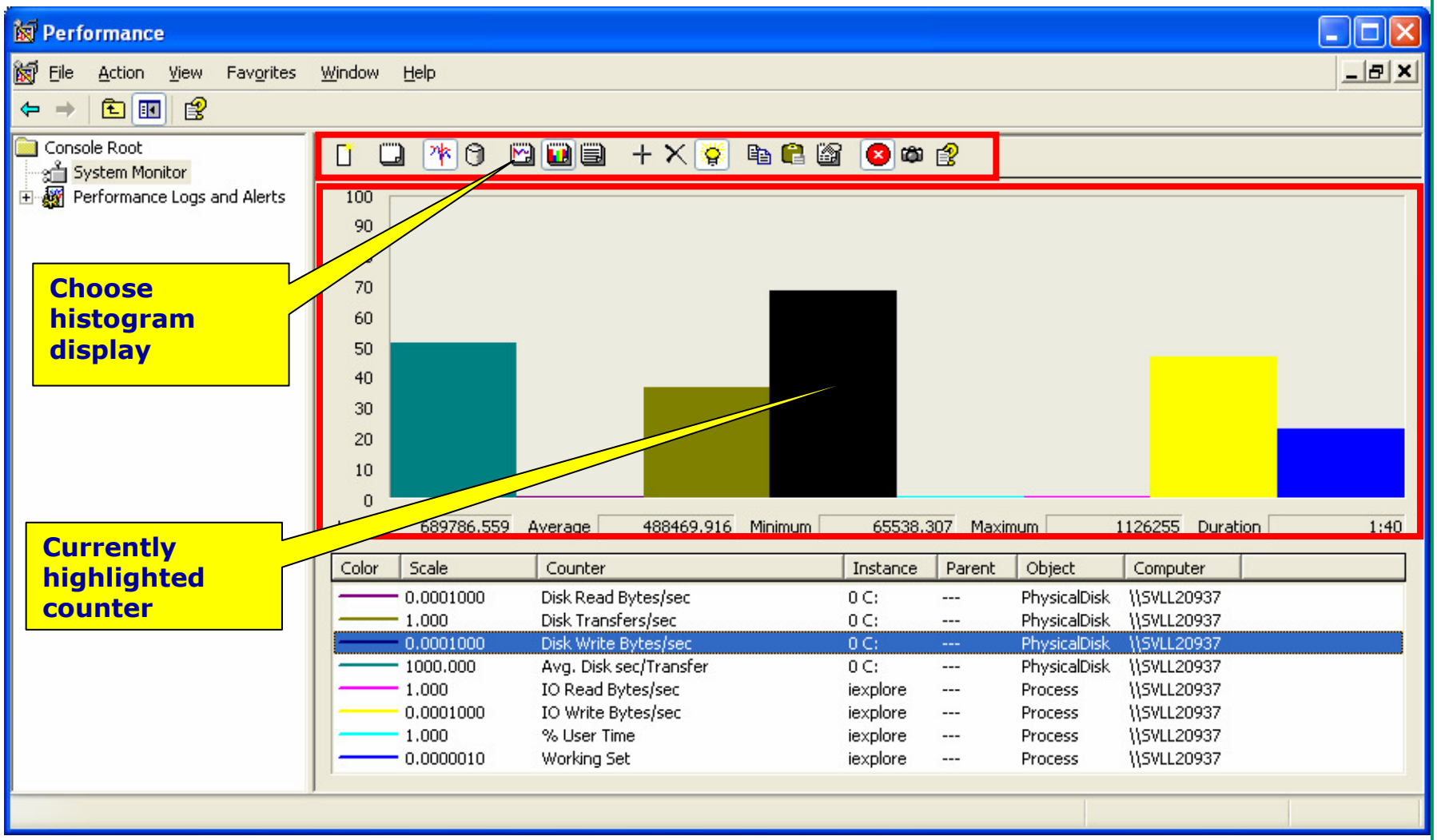
AMD
Smarter Choice

- **Primary tool used on Windows:**
- **Interactive, real-time GUI**
 - Graphical view
 - Histogram
 - Report (text)
 - Add counters from objects and watch them in real time
 - May require administrative or power user privileges
- **Example Counters:**
 - Cache Bytes
 - Avg. Disk Queue Length
 - Committed Bytes
 - % Processor Time
 - Pages/sec
 - Disk Write Bytes/sec
- **Example Objects:**
 - Memory
 - Physical Disk
 - Process
 - Processor
 - System

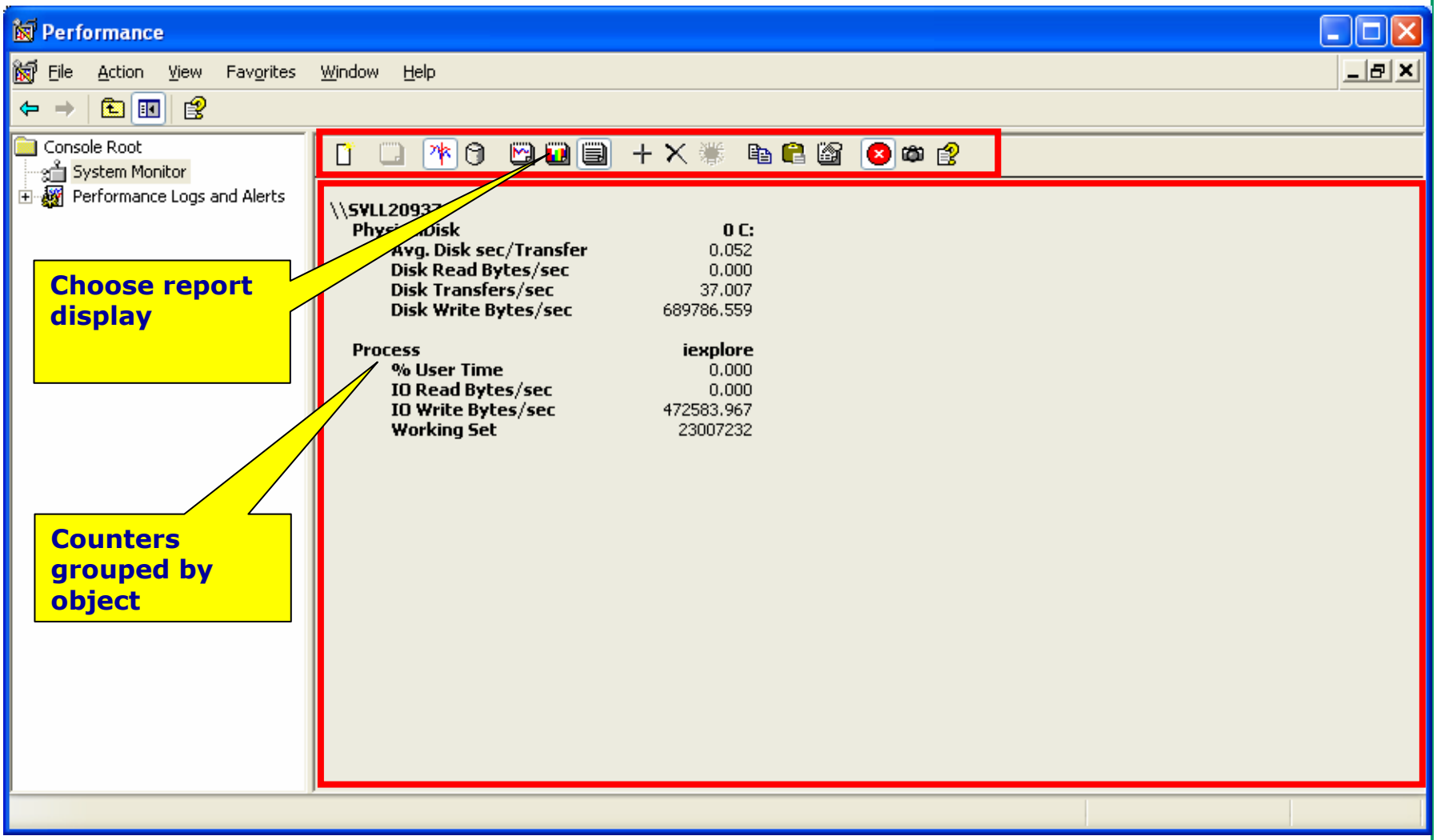
Performance Monitor Graphical View



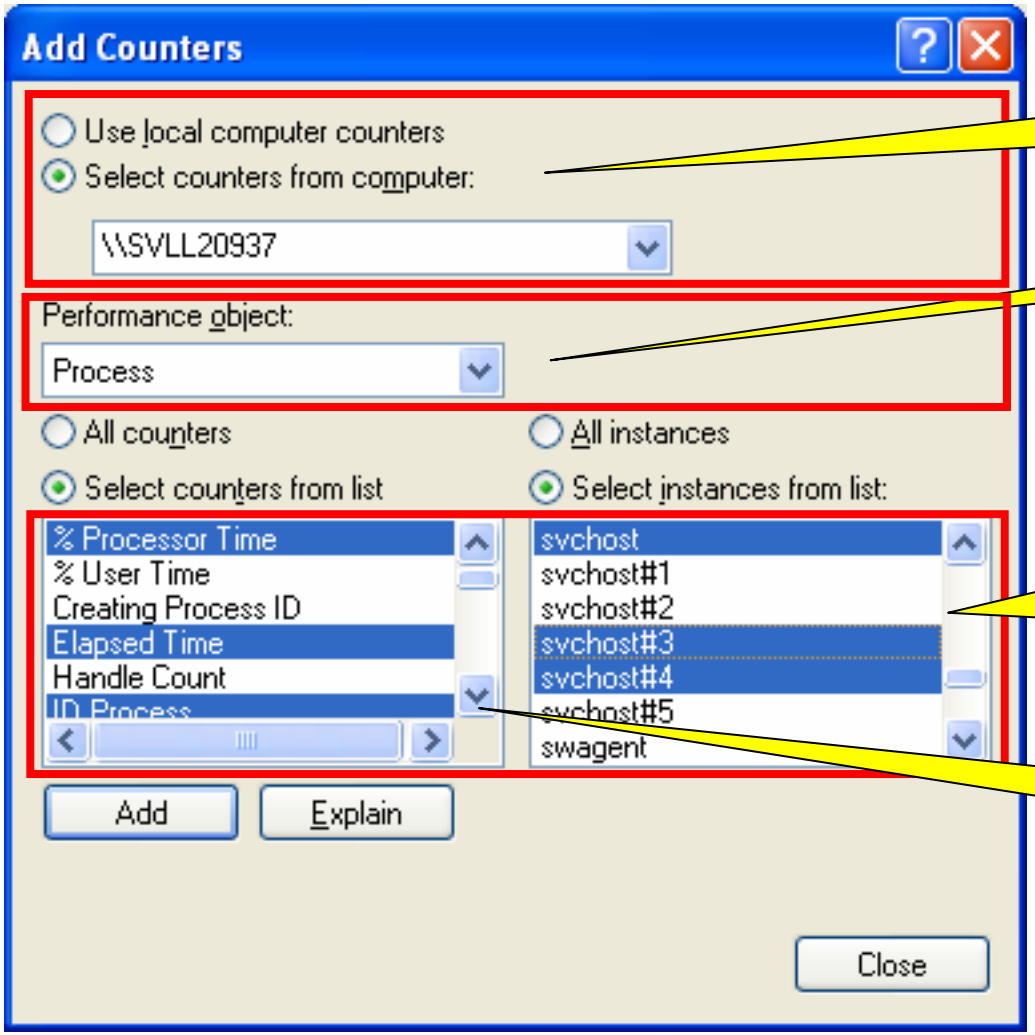
Performance Monitor Histogram



Performance Monitor Report Display



Add Counters Dialog



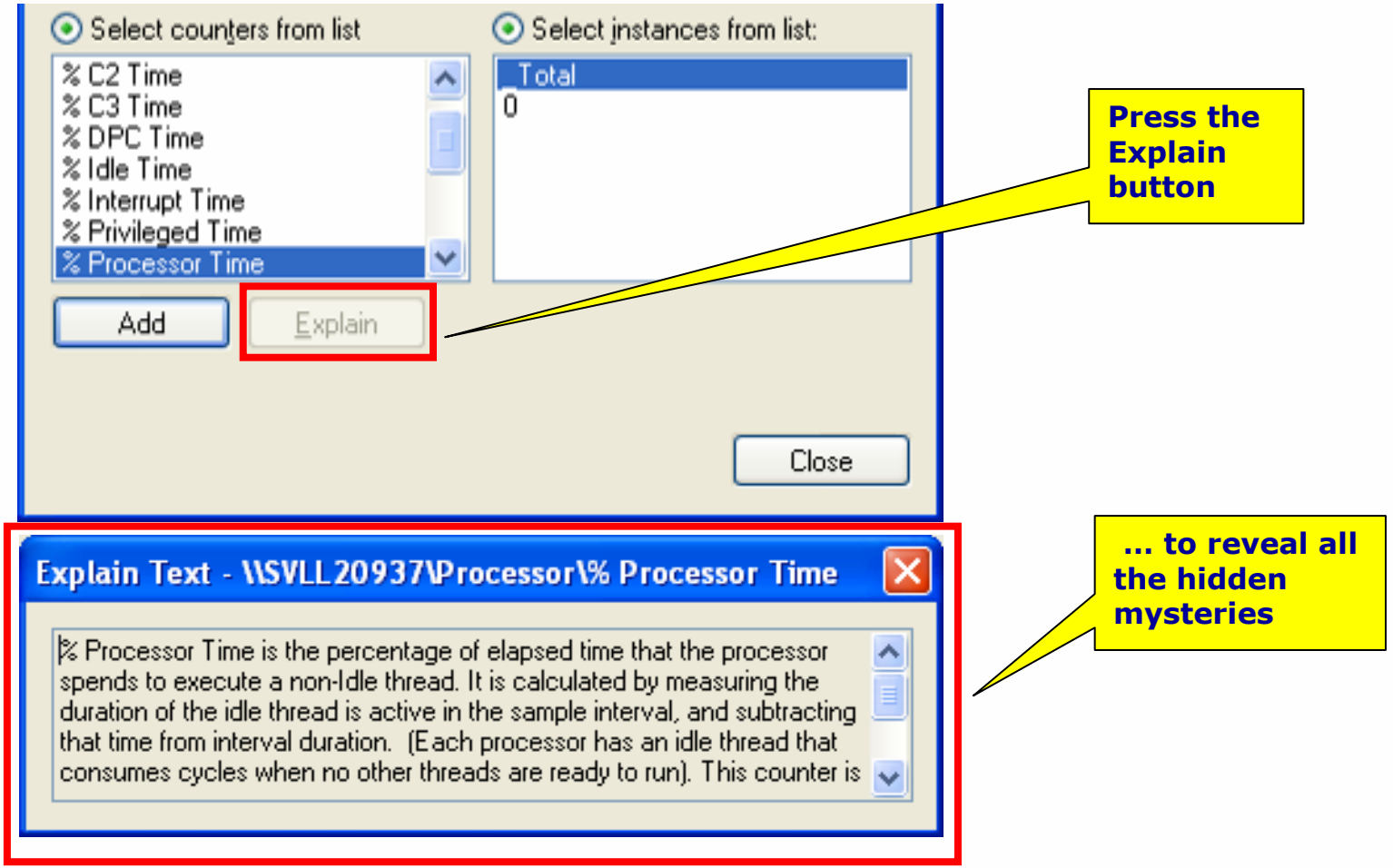
Monitor local or remote machines

Select object

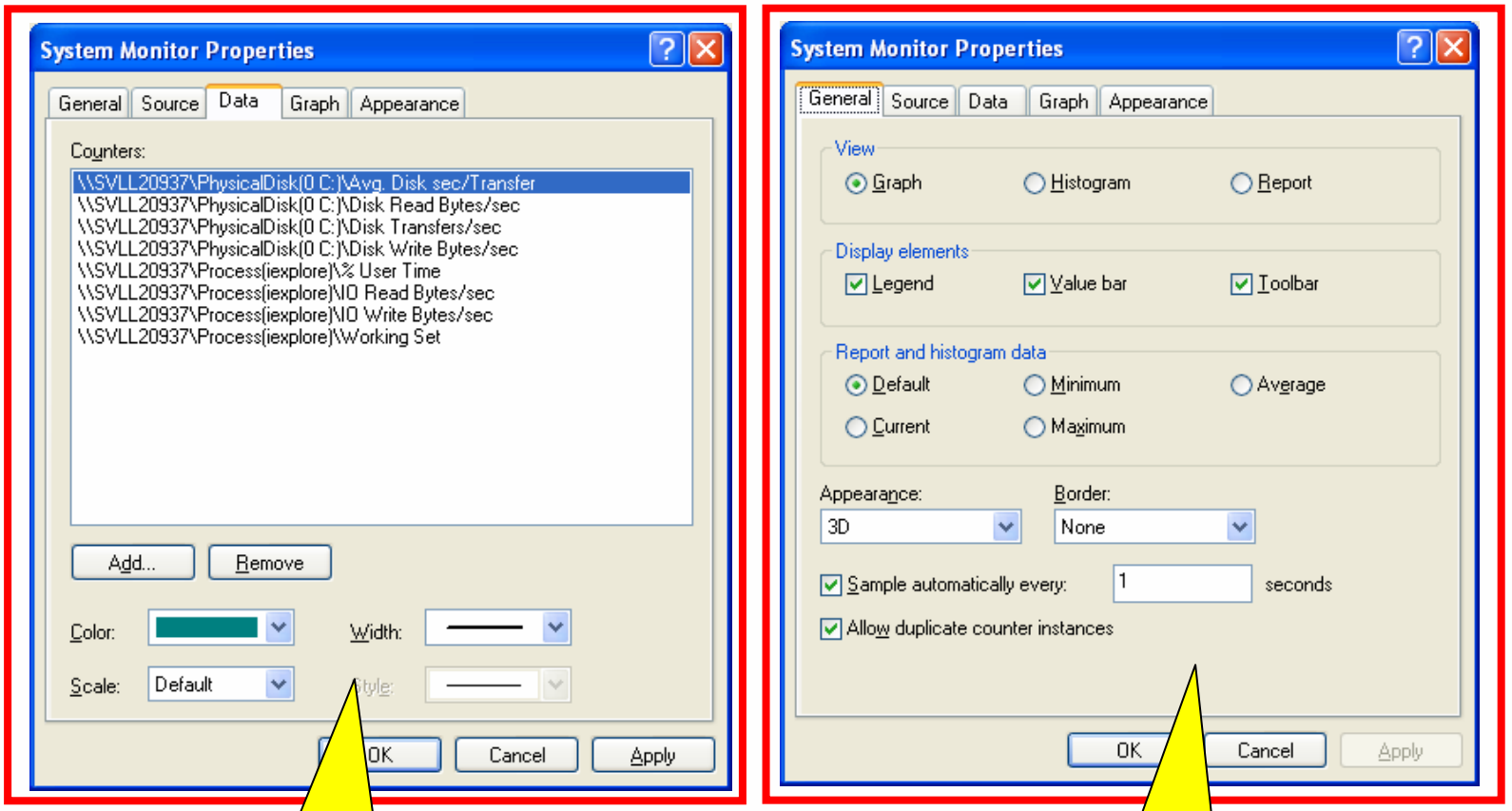
Use ctrl-click in listboxes to select individual counters and instances

Instances and process ids not tightly coupled

Add Counters Dialog



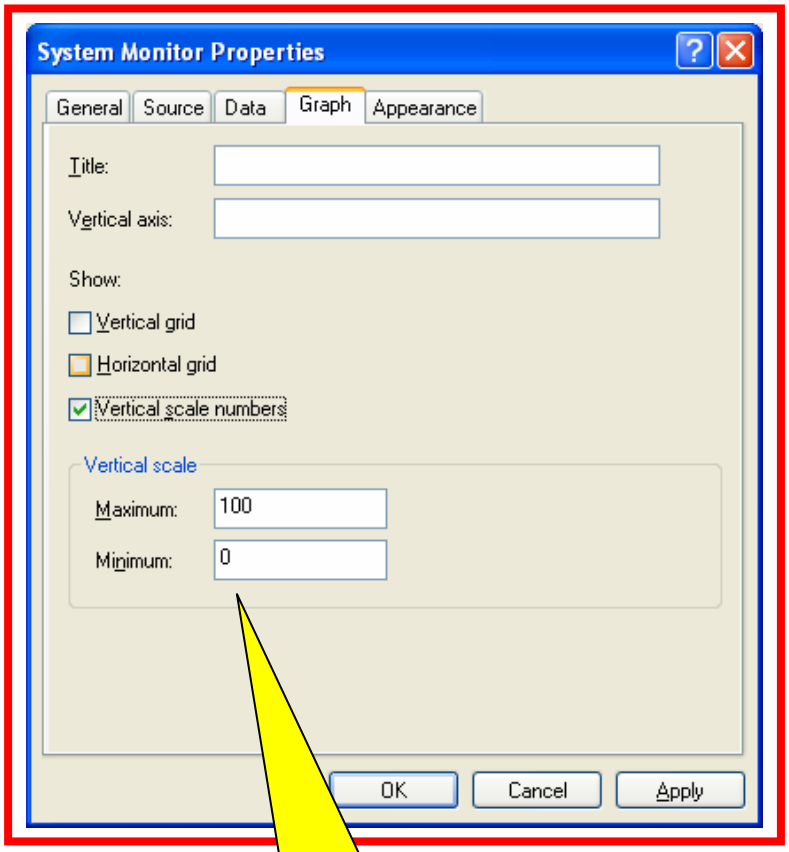
Properties Dialog – Data, General

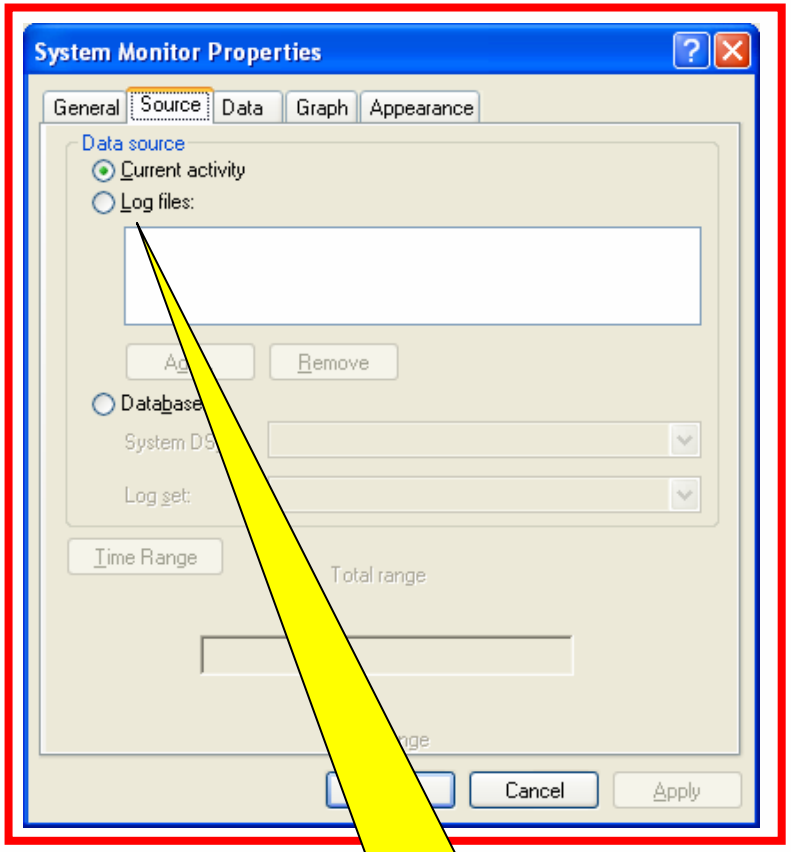


Data tab – set color, thickness, scale factor

General tab - other display properties, such as sample rate

Properties Dialog – Graph, Source

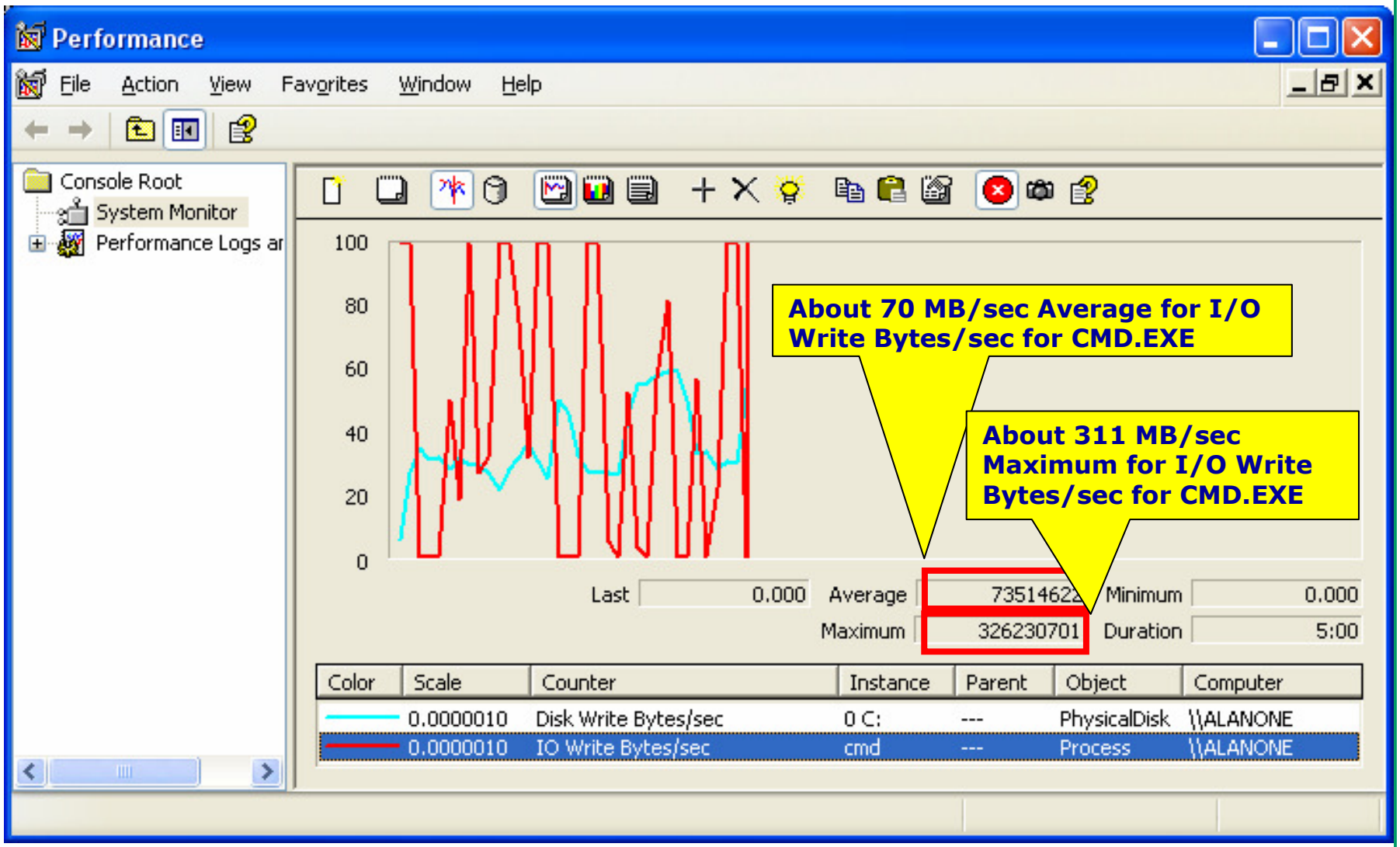




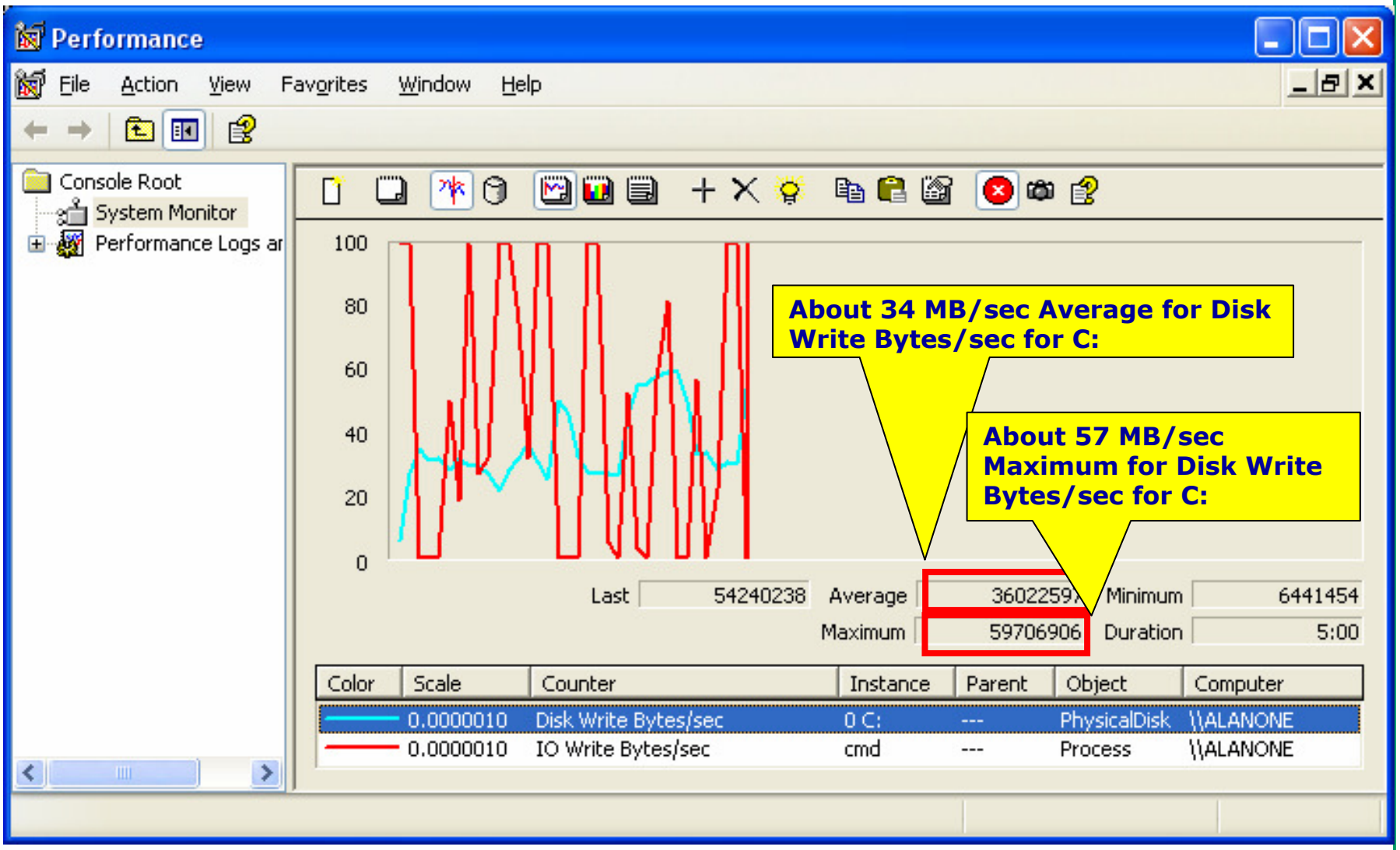
Graph tab - Changing vertical scale limit can be VERY useful.

Source tab - to analyze a logged session, use this.

Performance Monitor Example – I/O



Performance Monitor Example – I/O

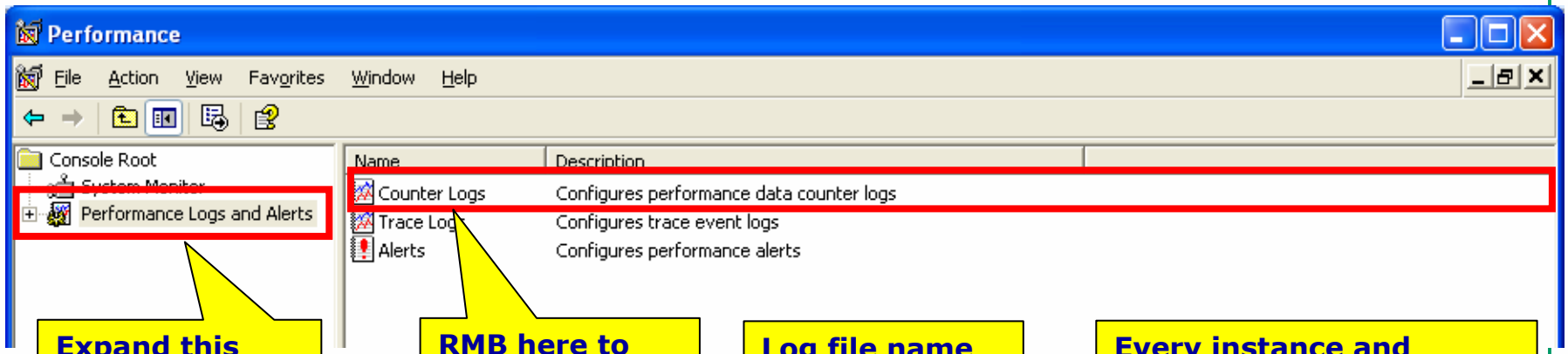


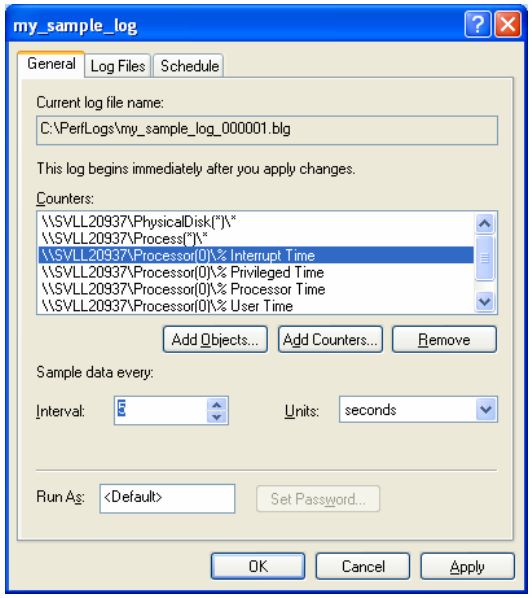
Performance Monitor Logs

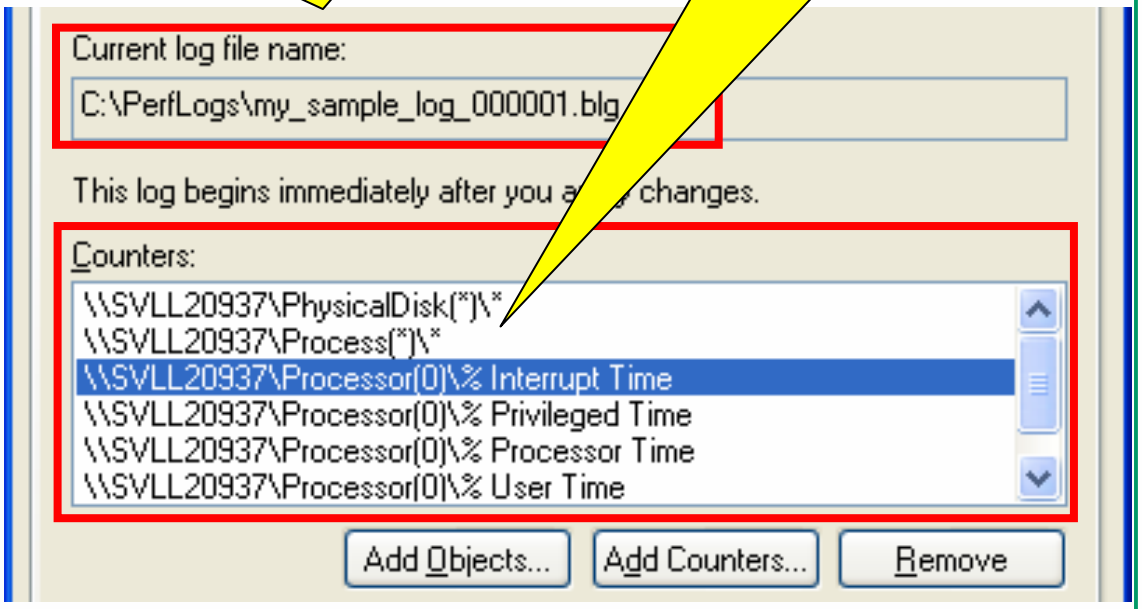
AMD
Smarter Choice

- **Create a collection that you can analyze later**
- **Administer with Perfmon or Logman**
- **Save a console view**
 - Customize view of your favorite counters
- **LOGMAN**
 - Command Line Tool
 - Start, stop logs
 - Add counters
- **Can log Event Traces as well**
- **Can set up Performance Alerts which get logged to Event Viewer**

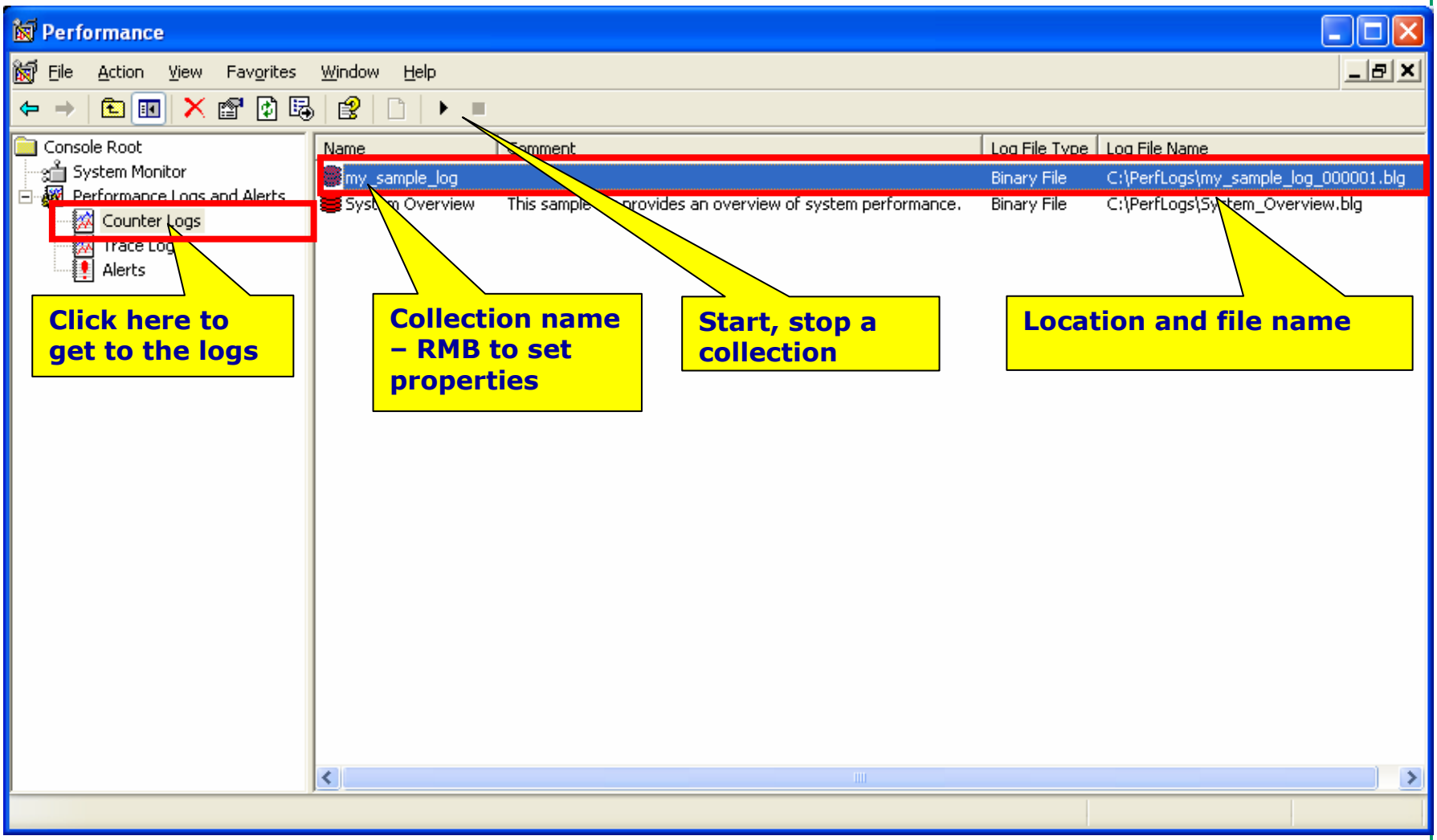
Performance Monitor Logs





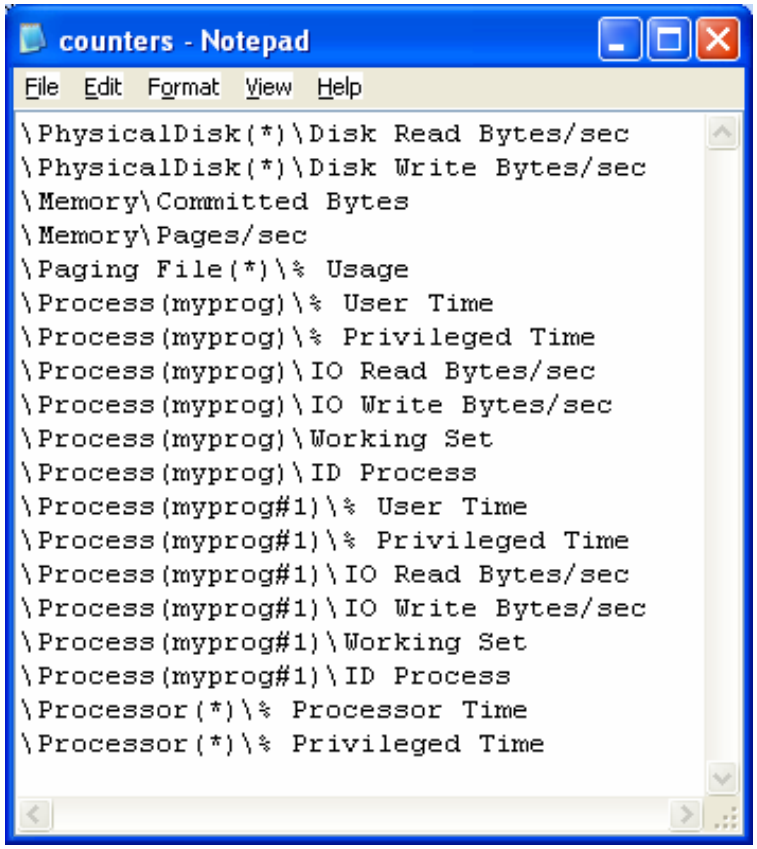


Performance Monitor Logs



Windows LOGMAN

- **Command Line Interface**
- **Good for monitoring specific batch jobs**
- **Applications Tab**
 - Add counters
 - Start
 - Stop
 - Collection interval
 - File location
- **Also Event Traces**
- **Other utilities**
 - RELOG
 - TYPEPERF
 - TRACERPT
 WMI Event Tracing



Windows LOGMAN

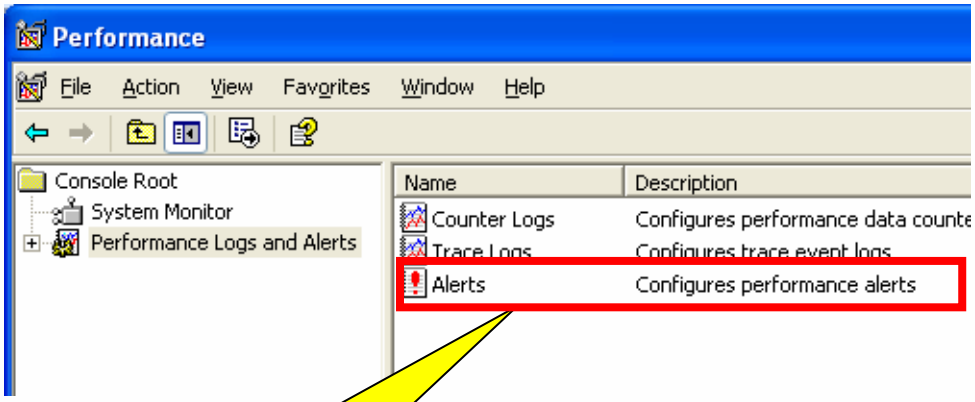
AMD Smarter Choice

- REM create binary log
- logman create counter MyCtrs -m start
- logman update counter MyCtrs -si 15
- logman update counter MyCtrs -o c:\PerfLogs\MyCtrs
- logman update counter MyCtrs -cf c:\PerfLogs\counters.txt
- logman update counter MyCtrs -f bin

- logman start counter MyCtrs
- REM now run your batch job
- logman stop counter MyCtrs

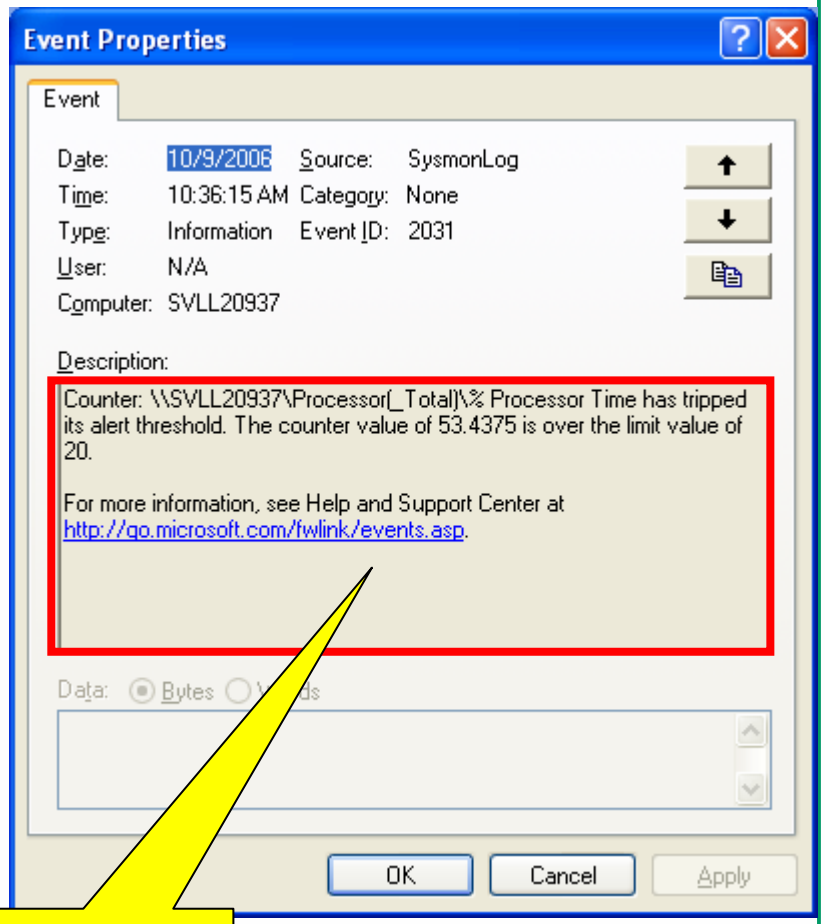
- REM create CSV file
- relog -f CSV myctrs_000001.blg -o myctrs.csv -q

Performance Alerts



RMB here to configure Alerts

- Add counters with a threshold, which will create events in the Event Log



Description of counter vs. threshold

Suggested Disk Counters

AMD
Smarter Choice

- Physical Disk\Avg. Disk Queue Length
- Physical Disk\Current Disk Queue Length
- Physical Disk\Disk Read Bytes/sec
- Physical Disk\Disk Write Bytes/sec
- Physical Disk\Disk Transfers/sec
- Physical Disk\Avg. Disk sec/Transfer

Other Suggested Counters

AMD
Smarter Choice

- Memory\Available MBytes
- Memory\Committed Bytes
- Memory\Pages/sec

- Paging File\% Usage

- Processor\% Processor Time
- Processor\% Privileged Time

- Process\% User Time
- Process\% Privileged Time
- Process\IO Read Bytes/sec
- Process\IO Write Bytes/sec
- Process\Working Set

Key tools under Linux

AMD Smarter Choice

top command

The top program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of tasks currently being managed by the Linux kernel. You can diagnose 70% of your grief this way. Type the top command:

```
$ top
top - 12:59:19 up  4:37,  4 users,  load average: 0.33, 0.23, 0.18
Tasks: 120 total,   1 running, 119 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.9% us,  0.2% sy,  0.0% ni, 98.5% id,  0.3% wa,  0.0% hi,  0.1% si
Mem:   8172632k total,  1777700k used,  6394932k free,   118868k buffers
Swap:  2104472k total,        0k used,  2104472k free,  1297716k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2089 root      15   0  9680 1268  884 R    2  0.0   0:00.01 top
    1 root      16   0   776  304  252 S    0  0.0   0:01.66 init
    2 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S    0  0.0   0:00.06 migration/1
    5 root      34  19     0    0    0 S    0  0.0   0:00.02 ksoftirqd/1
    6 root      RT   0     0    0    0 S    0  0.0   0:00.02 migration/2
    7 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/2
    8 root      RT   0     0    0    0 S    0  0.0   0:00.04 migration/3
    9 root      34  19     0    0    0 S    0  0.0   0:00.04 ksoftirqd/3
```

Key tools under Linux

AMD
Smarter Choice

sar command

The sar command is a performance monitoring tool. It can find out what Linux is doing all the time. It can generate report and email them to sys admin. In order to use sar you need to install sysstat package, since sar is not available by default on all distributions. sar displays performance data for:

- CPU and Queue
- Disk I/O
- Swap and Memory
- CPU interrupts, Networking and much more

Key tools under Linux

AMD Smarter Choice

sar command

sar –u 2 5

Report CPU utilization for each 2 seconds. 5 lines are displayed.

sar –I 14 –o int14.file 2 10

Report statistics on IRQ 14 for each 2 seconds. 10 lines are displayed. Data are stored in a file called *int14.file*.

sar –r –n DEV –f /var/log/sa/sa16

Display memory, swap space and network statistics saved in daily data file 'sa16'.

sar –A

Display all the statistics saved in current daily data file.

Key tools under Linux

AMD
Smarter Choice

vmstat command

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity. The first report produced gives averages since the last reboot. Additional reports give information on a sampling period of length *delay*. The process and memory reports are instantaneous in either case.

Key tools under Linux

AMD Smarter Choice

vmstat command

vmstat [**-a**] [**-n**] [*delay* [*count*]]
vmstat [**-f**] [**-s**] [**-m**]
vmstat [**-S unit**]
vmstat [**-d**]
vmstat [**-p disk partition**]
vmstat [**-V**]

The **-a** switch displays active/inactive memory, given a 2.5.41 kernel or better.

The **-f** switch displays the number of forks since boot. This includes the fork, vfork, and clone system calls, and is equivalent to the total number of tasks created. Each process is represented by one or more tasks, depending on thread usage. This display does not repeat.

The **-m** displays slabinfo.

The **-n** switch causes the header to be displayed only once rather than periodically.

The **-s** switch displays a table of various event counters and memory statistics. This display does not repeat. *delay* is the delay between updates in seconds. If no delay is specified, only one report is printed with the average values since boot. *count* is the number of updates. If no count is specified and delay is defined, *count* defaults to infinity.

The **-d** reports disk statistics (2.5.70 or above required)

The **-p** followed by some partition name for detailed statistics (2.5.70or above required). The **-S** followed by k or K or m or M switches outputs between 1000,1024, 1000000, or 1048576 bytes

The **-V** switch results in displaying version information.

Key tools under Linux

AMD
Smarter Choice

iostat command

The **iostat** command is used for monitoring system input/output device loading by observing the time the devices are active in relation to their average transfer rates. The **iostat** command generates reports that can be used to change system configuration to better balance the input/output load between physical disks.

iostat Display a single history since boot report for all CPU and Devices.

iostat -d 2 Display a continuous device report at two second intervals.

iostat -d 2 6 Display six reports at two second intervals for all devices.

iostat -x hda hdb 2 6 Display six reports of extended statistics at two second intervals for devices hda and hdb.

iostat -p sda 2 6 Display six reports at two second intervals for device sda and all its partitions (sda1, etc.)

Key tools under Linux

AMD Smarter Choice

free command

free displays the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel. The shared memory column should be ignored; it is obsolete.

free [**-b** | **-k** | **-m**] [**-o**] [**-s** *delay*] [**-t**] [**-V**]

-b	The -b switch displays the amount of memory in bytes; the -k switch (set by default) displays it in kilobytes; the -m switch displays it in megabytes.
-t	The -t switch displays a line containing the totals.
-o	The -o switch disables the display of a "buffer adjusted" line. If the -o option is not specified, free subtracts buffer memory from the used memory and adds it to the free memory reported.
-s	The -s switch activates continuous polling delay seconds apart. You may actually specify any floating point number for delay, usleep(3) is used for microsecond resolution delay times.
-V	The -V displays version information.

Key tools under Linux

AMD
Smarter Choice

strace command

strace runs the specified *command* until it exits. It intercepts and records the system calls which are called by a process and the signals which are received by a process. The name of each system call, its arguments and its return value are printed on standard error or to the file specified with the **-o** option. **strace** is a useful diagnostic, instructional, and debugging tool. Each line in the trace contains the system call name, followed by its arguments in parentheses and its return value.

```
find . –print | xargs wc
strace –F –p 28240
```

…Produces…

Key tools under Linux

AMD Smarter Choice

strace command

```
find . -print | xargs wc
strace -F -p 28240
 fchdir(5)                                = 0
 close(5)                                 = 0
 open("..", O_RDONLY|O_NOFOLLOW)          = 5
 fchdir(5)                                = 0
 close(5)                                 = 0
 lstat("opcontrol.1.in", {st_mode=S_IFREG|0644, st_size=2992, ...}) = 0
 lstat("Makefile", {st_mode=S_IFREG|0644, st_size=16819, ...}) = 0
 lstat("internals.html", {st_mode=S_IFREG|0644, st_size=39346, ...}) = 0
 lstat("internals.xml", {st_mode=S_IFREG|0644, st_size=26914, ...}) = 0
 lstat("opannotate.1.in", {st_mode=S_IFREG|0644, st_size=2683, ...}) = 0
 lstat("opreport.1", {st_mode=S_IFREG|0644, st_size=2731, ...}) = 0
 lstat("op_help.1", {st_mode=S_IFREG|0644, st_size=931, ...}) = 0
 lstat("opgprof.1.in", {st_mode=S_IFREG|0644, st_size=1113, ...}) = 0
 lstat("srcdoc", {st_mode=S_IFDIR|0755, st_size=152, ...}) = 0
 open("srcdoc", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = 5
 fstat(5, {st_mode=S_IFDIR|0755, st_size=152, ...}) = 0
 fcntl(5, F_SETFD, FD_CLOEXEC)            = 0
 getdents64(5, /* 6 entries */, 4096)     = 168
 getdents64(5, /* 0 entries */, 4096)     = 0
 close(5)                                 = 0
 open("srcdoc", O_RDONLY|O_NOFOLLOW)      = 5
 fchdir(5)                                = 0
```

Key tools under Solaris

AMD Smarter Choice

truss command

truss executes the specified command and produces a trace of the system calls it performs, the signals it receives, and the machine faults it incurs. Each line of the trace output reports either the fault or signal name or the system call name with its arguments and return value(s).

```
# trace the find command
truss find . -print >find.out
# trace the find command, but only for opens, closes, reads, and
   writes
truss -t open,close,read,write find . -print >find.out
# track all usage of libc
truss -u libc
# Don't track libc, but everything else.
truss -u '*' -u !libc
```

Key tools under Solaris

AMD Smarter Choice

DTrace command

DTrace A dynamic tracing facility that provides a comprehensive view of operating system and application behaviour. It has functionality similar to truss, apptrace, prex and mdb, bundled into a single scriptable tool that can examine both userland activity and the kernel. DTrace can be used on live production servers with often negligible impact on performance. This is a new feature in Solaris 10.

```
# disk io provider events
# dtrace -ln 'io:genunix::'
   ID     PROVIDER       MODULE            FUNCTION NAME
9571           io      genunix             biodone done
9572           io      genunix             biowait wait-done
9573           io      genunix             biowait wait-start
# Disk size by process
dtrace -n 'io:::start { printf("%d %s %d",pid,execname,args[0]->b_bcount); }'

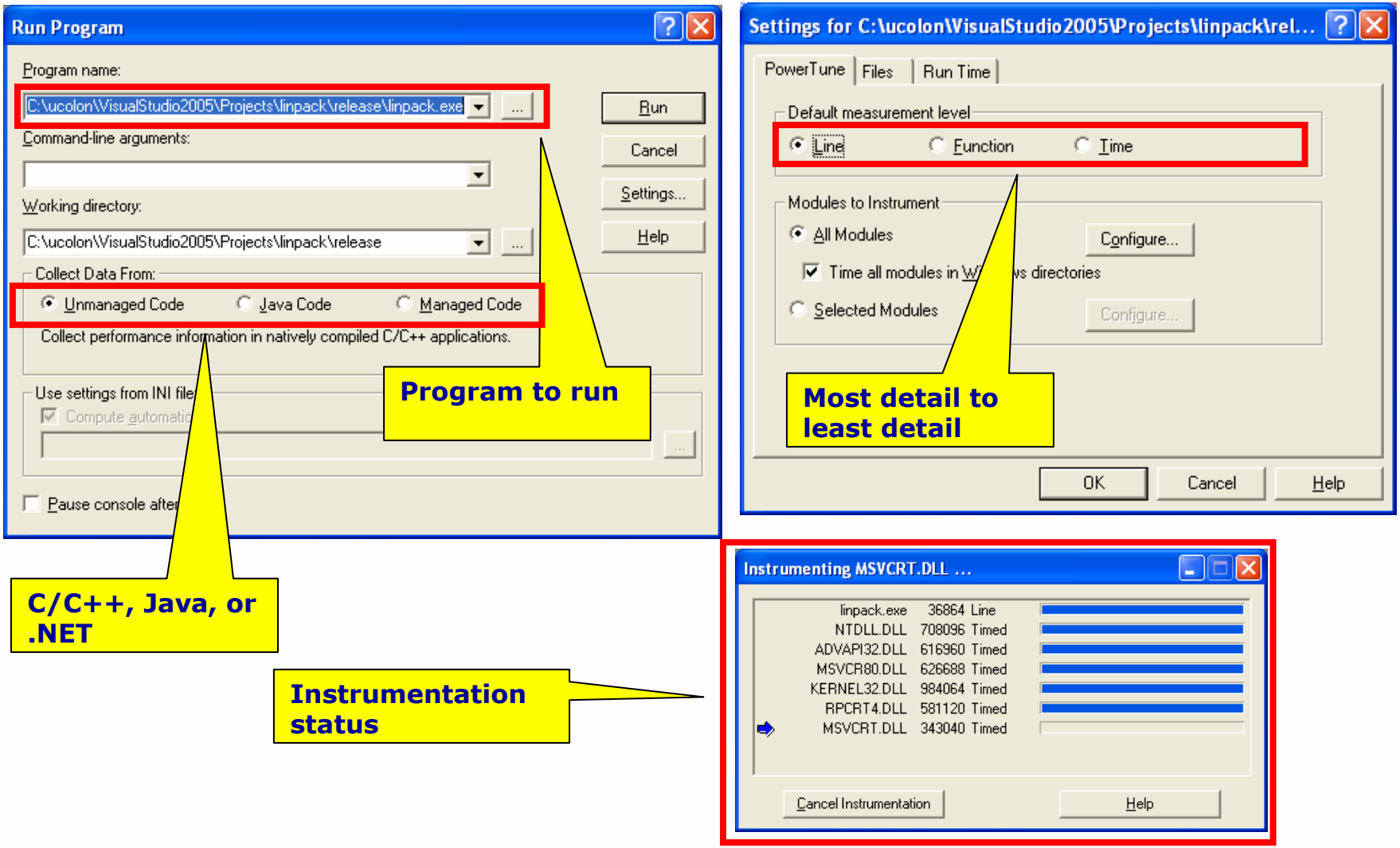
# Pages paged in by process
dtrace -n 'vminfo:::pgpgin { @pg[execname] = sum(arg0); }'
```

IBM Rational Quantify for Windows

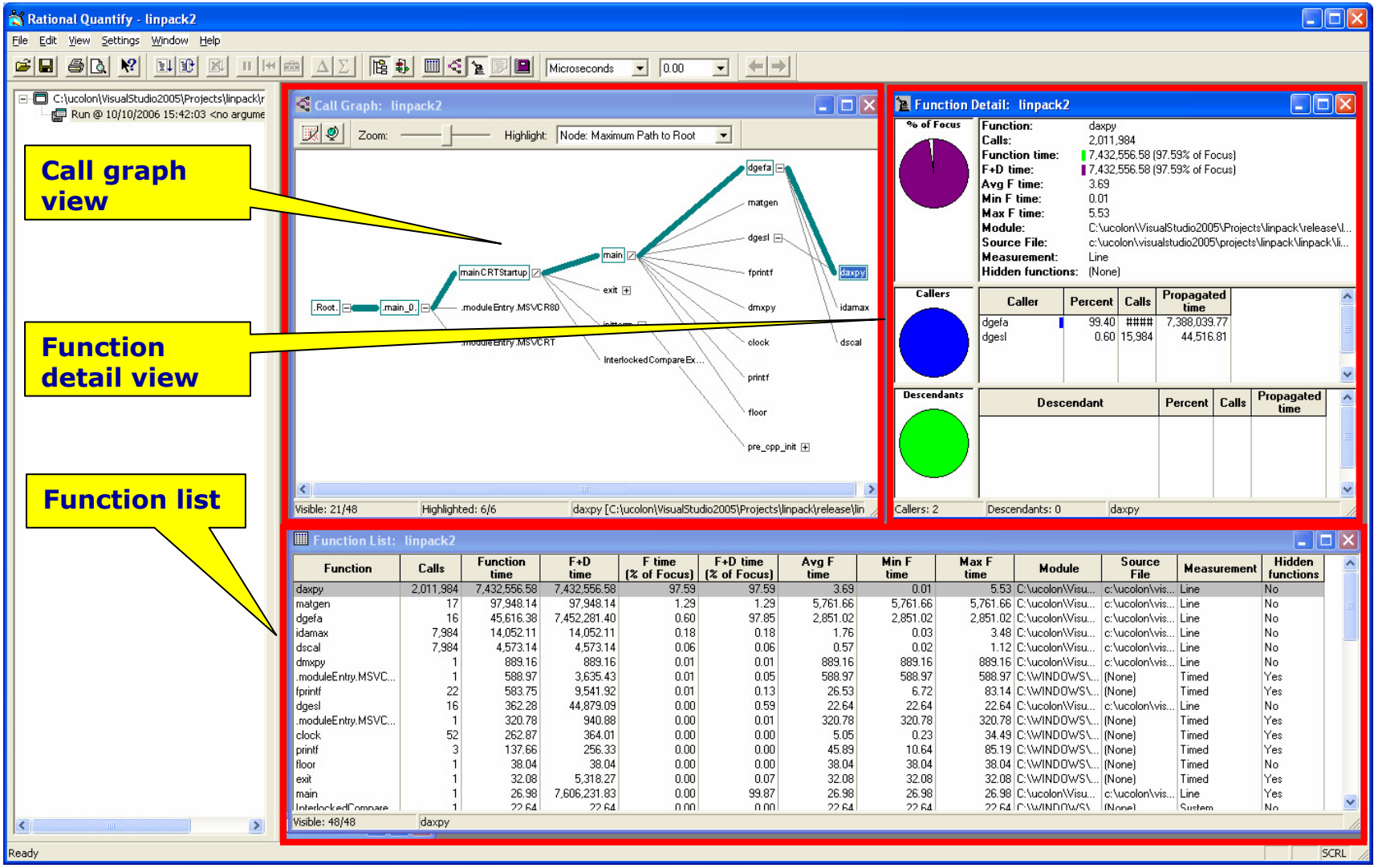
AMD Smarter Choice

- **Instrumented Source Level Profiler**
- **Profiling**
 - Line-based, most detailed
 - Function-based
 - Time-based, exported functions only
 - Pre-instrumented available
 - C/C++, Java Code, .NET Managed Code (C#, VB.NET)
 - 32-bit only for Windows
 - Link with /FIXED:NO
- **Tool sets**
 - MS VC++ 6, MS VC++ .NET,
 - Visual Studio .NET - ASP.NET, Visual Basic, C#, J#
 - IBM and Sun JVMs
- **Also**
 - Purify
 - PureCoverage

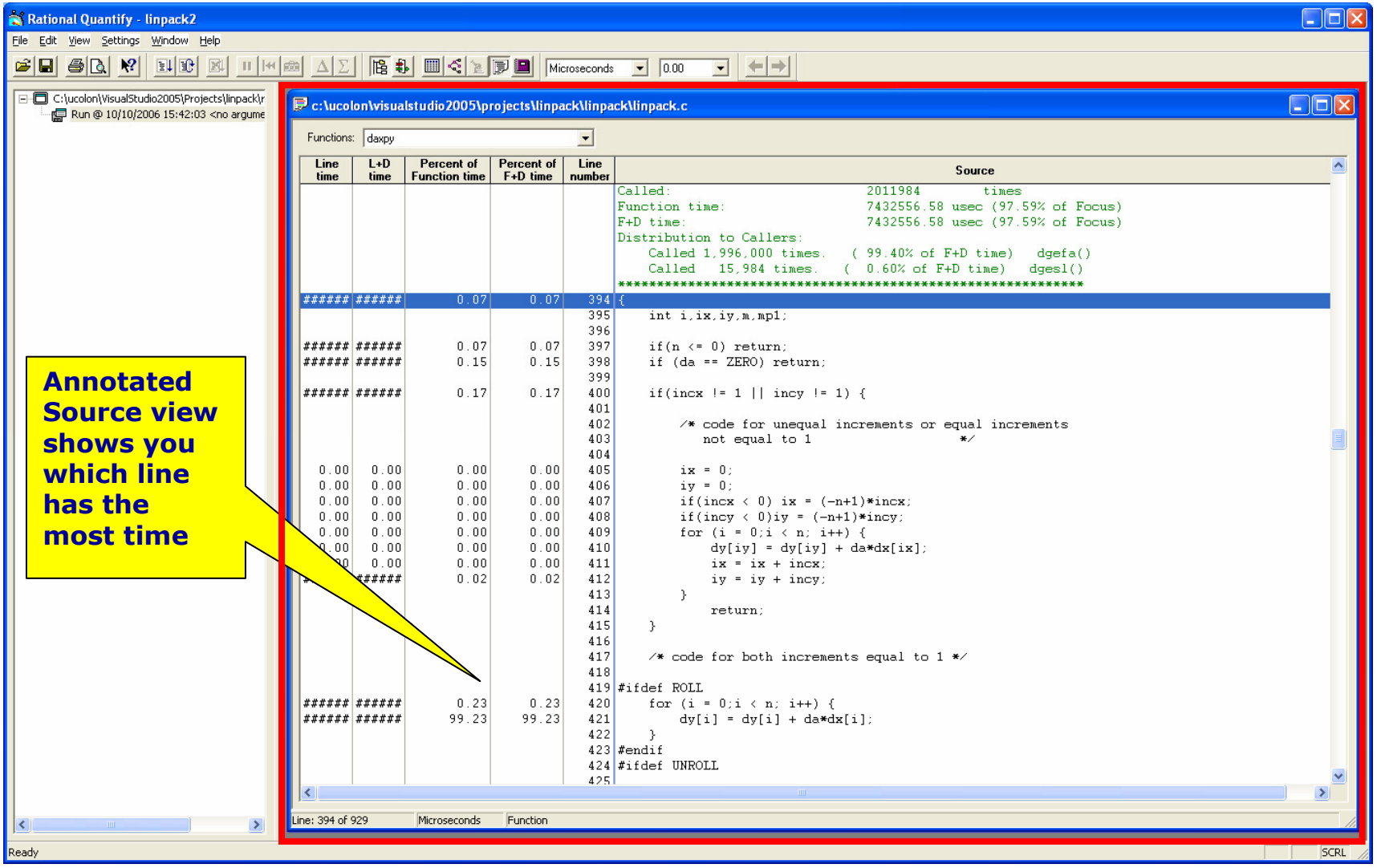
Quantify Session



Quantify Session



Quantify Session

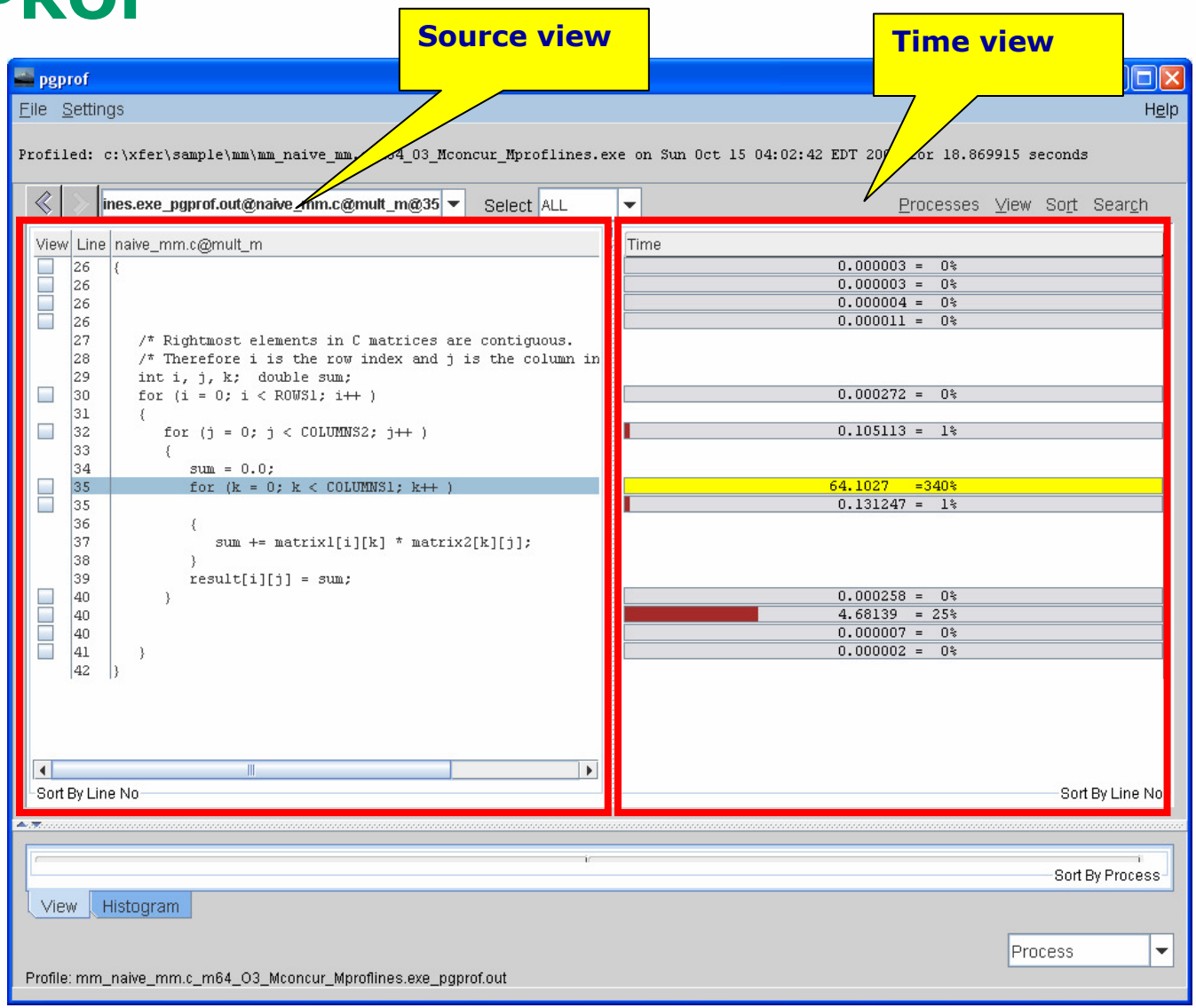


PGPROF and PGDBG (Windows & Linux)

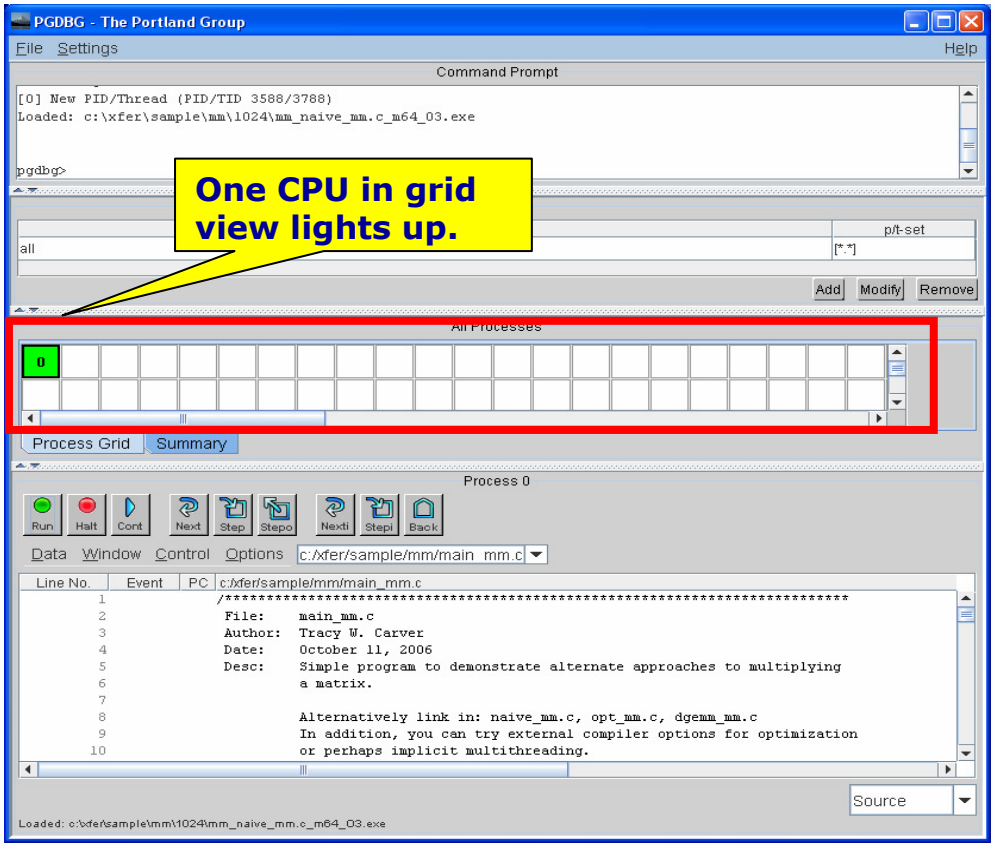
- **PGPROF - Instrumented Source Level Profiler**
- **Profiling**
 - Line-based, most detailed
 - Function-based, least detailed
- **Tool sets**
 - PGI Compiler

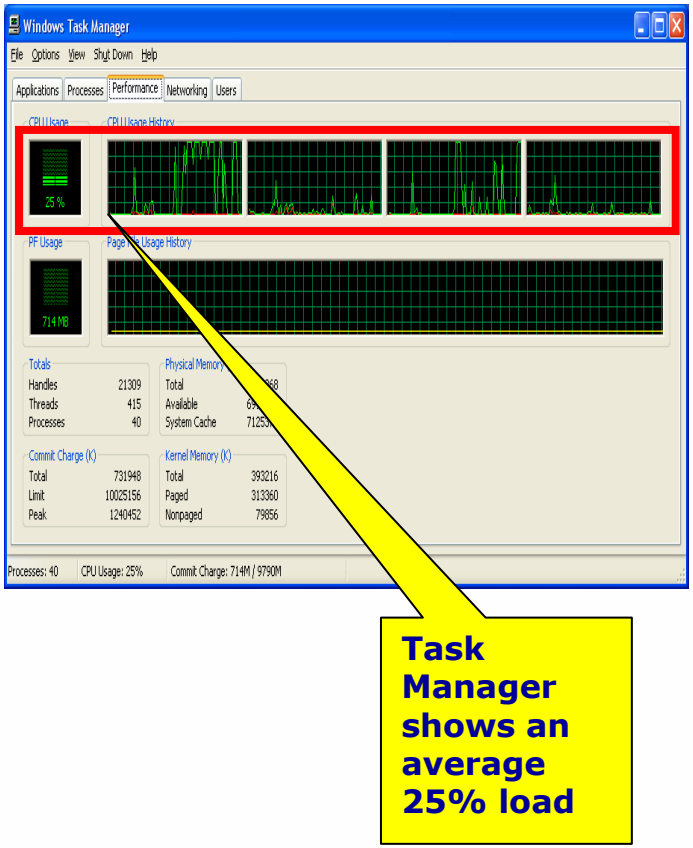
- **PGDBG – parallel debugger**
- **OpenMP thread-parallel**
 - PGI Workstation and PGI Server
- **MPI parallel**
 - PGI CDK Cluster Development Kit

PGPROF

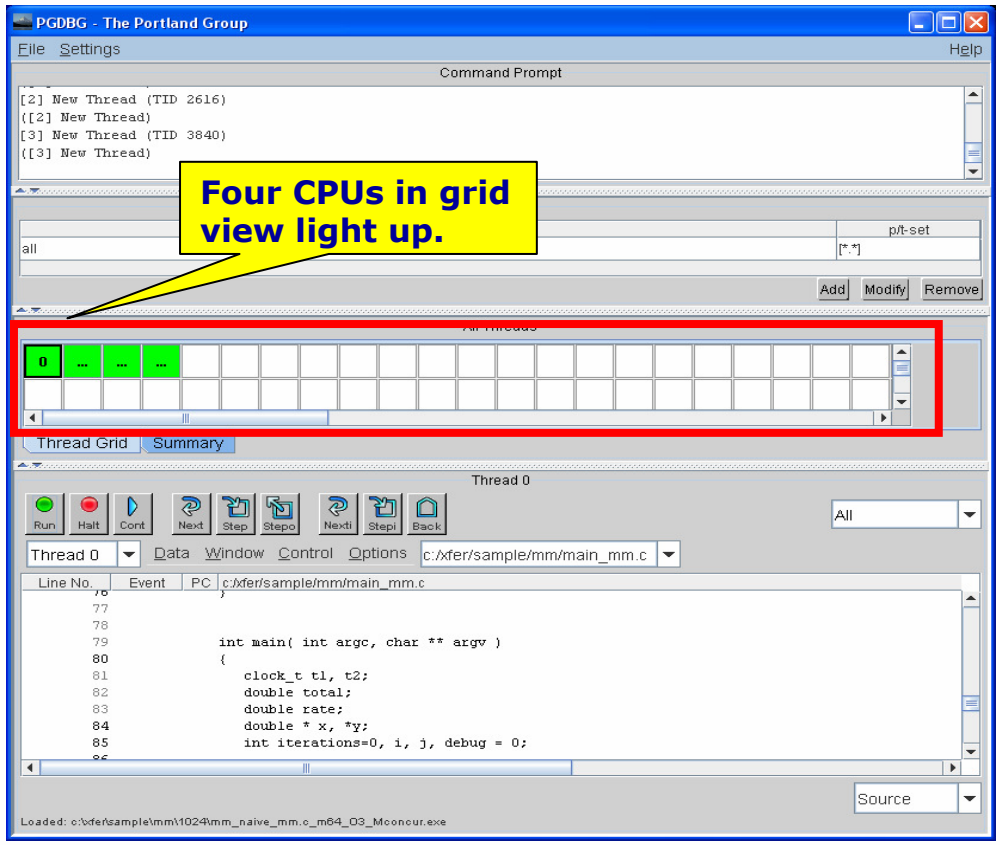

Source view
Time view
pgprof
File Settings
Help
Profiled: c:\xfer\sample\mm\mm_naive_mm.c_m64_O3_Mconcur_Mproflines.exe on Sun Oct 15 04:02:42 EDT 200 for 18.869915 seconds
ines.exe_pgprof.out@naive_mm.c@mult_m@35
Select ALL
Processes View Sort Search
View Line naive_mm.c@mult_m
Time
26 { 0.000003 = 0%
26 0.000003 = 0%
26 0.000004 = 0%
26 0.000011 = 0%
27 /* Rightmost elements in C matrices are contiguous.
28 /* Therefore i is the row index and j is the column in
29 int i, j, k; double sum;
30 for (i = 0; i < ROWS1; i++) 0.000272 = 0%
31 {
32 for (j = 0; j < COLUMNS2; j++) 0.105113 = 1%
33 {
34 sum = 0.0;
35 for (k = 0; k < COLUMNS1; k++) 64.1027 =340%
35 0.131247 = 1%
36 {
37 sum += matrix1[i][k] * matrix2[k][j];
38 }
39 result[i][j] = sum;
40 } 0.000258 = 0%
40 4.68139 = 25%
40 0.000007 = 0%
41 } 0.000002 = 0%
42 }
Sort By Line No
Sort By Line No
Sort By Process
View Histogram
Process
Profile: mm_naive_mm.c_m64_O3_Mconcur_Mproflines.exe_pgprof.out

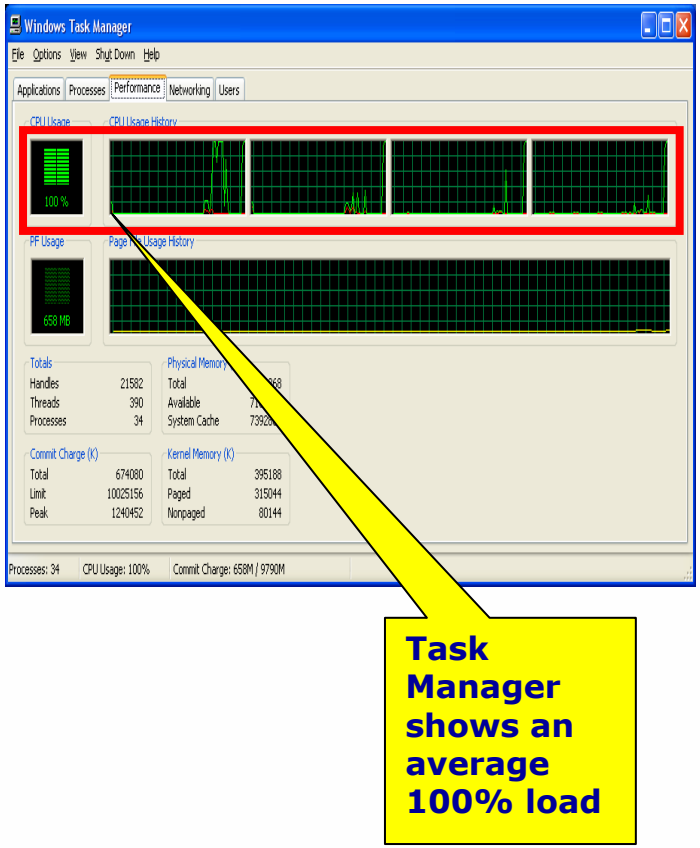
PGDBG – one thread running





PGDBG – 4 threads running with -Mconcur





OProfile Introduction

AMD Smarter Choice

- OProfile is a system-wide profiler for Linux
- Tells you where you are spending time in code
- Details
 - Sample-based profiler
 Less interference with your program
 - Generally requires root privileges
 - Source and documentation
 http://oprofile.sourceforge.net
- AMD CodeAnalyst uses its own version of oprofile

OProfile tools

AMD
Smarter Choice

- OProfile daemon
 - runs in the background
 - collects data from performance counter registers
- opcontrol
 - Configures background daemon
 - Starts and stop sampling
- opreport and opannotate
 - used to analyze the collected data

OProfile tools and process

AMD
Smarter Choice

- opcontrol

Configure and start daemon
Start sampling
Start workload
Stop workload
Stop sampling

- opreport / opannotate

Generate report

- Write wrapper scripts to control oprofile

OProfile events

AMD
Smarter Choice

- Analyze processor-specific events involving
 - processor cycles
 - translation look-aside buffer (TLB) misses
 - memory references
 - branch mispredictions
 - cache misses
 - interrupt handlers
 - List events with opcontrol --list-events

OProfile events

AMD Smarter Choice

- Sample count
 - how often counter is incremented for each event
- Default event
 - CPU_CLK_UNHALTED
 - counts processor clock cycles in unhalted state
- Select clock cycles with sample count of 3000, and cache misses with sample count of 500

```
opcontrol --event=CPU_CLK_UNHALTED:3000:0:1:1 \
--event=DATA_CACHE_MISSES:500:0:1:1
```

OPcontrol basic command line options

AMD Smarter Choice

opcontrol option	description
--list-events	List processor events and unit masks
--vmlinux=<kernel image>	The kernel image file that will be profiled
--no-vmlinux	Do not profile the kernel
--reset	Clears data from current session
--setup	Sets up the daemon before running it
--event=<processor event>	Monitor the given processor event
--start	Start sampling
--dump	Flush the data to the daemon
--stop	Stop data sampling
--shutdown	Kill the daemon

OPcontrol basic command line options

AMD Smarter Choice

opcontrol option	description
--save=sessionname	Save data from current session to sessionname.
--status	Show configuration information.
--deinit	Shut down daemon. Unload the oprofile module and oprofilefs.

- Archive data with –save=sessionname
 - Data goes to /var/lib/oprofile/samples/sessionname
- Sometimes difficult to switch counters
 - Use --deinit to completely unload the daemon

OProfile examples

AMD
Smarter Choice

```
An example script:
#stop any previous session before starting
opcontrol --stop
opcontrol --dump
opcontrol --reset
opcontrol --start

./a.out -iterations $iterations -l $1

opcontrol --stop
opcontrol --dump
```

OProfile examples

AMD Smarter Choice

```
# now report the data

mkdir -p output/${testname}
opreport  -r  --long-filenames  > output/${testname}/report.txt
opannotate        --source -t 1 > output/${testname}/source.txt
opannotate        --assembly -t 1  >  output/${testname}/asm.txt
opannotate        --source --assembly -t 1  > \
                     output/${testname}/mixed.txt
# the save will move /var/lib/oprofile/samples/current to
# /var/lib/oprofile/samples/${testname}
opcontrol --save=${testname}
```

–t 1 option

- only report sample locations that take up at least one percent of the overall profile

OProfile examples

AMD Smarter Choice

In order to study cache misses, this command was used:

```
opcontrol --event=CPU_CLK_UNHALTED:3000:0:1:1 \
--event=DATA_CACHE_MISSES:500:0:1:1
```

This command can theoretically tell you what events are being profiled:

```
opcontrol --status
```

However, it was necessary to do this to ensure the daemon was collecting the right counters.

```
opcontrol -stop
opcontrol -dump
opcontrol --reset
opcontrol --deinit
opcontrol --event=CPU_CLK_UNHALTED:3000:0:1:1 \
--event=DATA_CACHE_MISSES:500:0:1:1
```

OProfile examples

AMD Smarter Choice

```
/*
 * Command line: opannotate --source -t 1
 *
 * Interpretation of command line:
 * Output annotated source file with samples
 * Output files where samples count reach 1% of the samples
 *
 * CPU: AMD64 processors, speed 1000 MHz (estimated)
 * Counted CPU_CLK_UNHALTED events (Cycles outside of halt state) with a unit mask of 0x00
    (No unit mask) count 3000
 * Counted DATA_CACHE_MISSES events (Data cache misses) with a unit mask of 0x00 (No unit
    mask) count 500
 */
/*
 * Total samples for file :
    "/home/tcarver/memb_matrix/inner_oprofile/inner_array_unrolled.c"
 *
 * 1182893 49.9675146923 81.5374
 */
```

OProfile examples

AMD Smarter Choice

```
                                    :double inner( double * x, double *y, int n )
                                    :{
                                    :   double sum = 0;
                                    :   int i, j, k;
                                    :
     4 1.7e-04     0        0      :   if (n<4) /* inner total: 1182893 49.9675146923 81.5374 */
                                    :   {
                                    :      for (i = 0; i < n; i++)
                                    :         sum +=  x[i] + y[i];
                                    :      return( sum );
                                    :   }
                                    :   j = 4* (n / 4);
                                    :   k = n % 4;
273093 11.5359 34664 19.2374        :   for ( i = 0; i < j; i+=4 )
                                    :   {
139265  5.8828 17817  9.8878        :      sum +=  x[i]   + y[i];
234875  9.9215 24365 13.5218        :      sum +=  x[i+1] + y[i+1];
203266  8.5863 27542 15.2849        :      sum +=  x[i+2] + y[i+2];
332326 14.0380 42531 23.6033        :      sum +=  x[i+3] + y[i+3];
                                    :   }
    59  0.0025     4  0.0022        :   for ( ; i < n; i++)
                                    :      sum += x[i] * y[i];
                                    :   return( sum );
     5 2.1e-04     0        0      :}
                                    :
```



AMD
Smarter Choice

Lunch &
Celoxica Demonstration

Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice

Agenda

AMD
Smarter Choice

9:00 Opening remarks

9:15 AMD Architecture

10:00 Coffee Break

10:15 Multi-Threading

11:15 Tools (Part 1)

12:00 Lunch and Celoxica demonstration

12:45 Tools (Part 2)

13:30 Compilers and Libraries

14:30 Coffee Break

14:45 AMD CodeAnalyst demo

15:15 Q & A

16:00 Closing remarks



AMD
Smarter Choice

Tools (Part 2)
Performance Analysis with AMD CodeAnalyst™

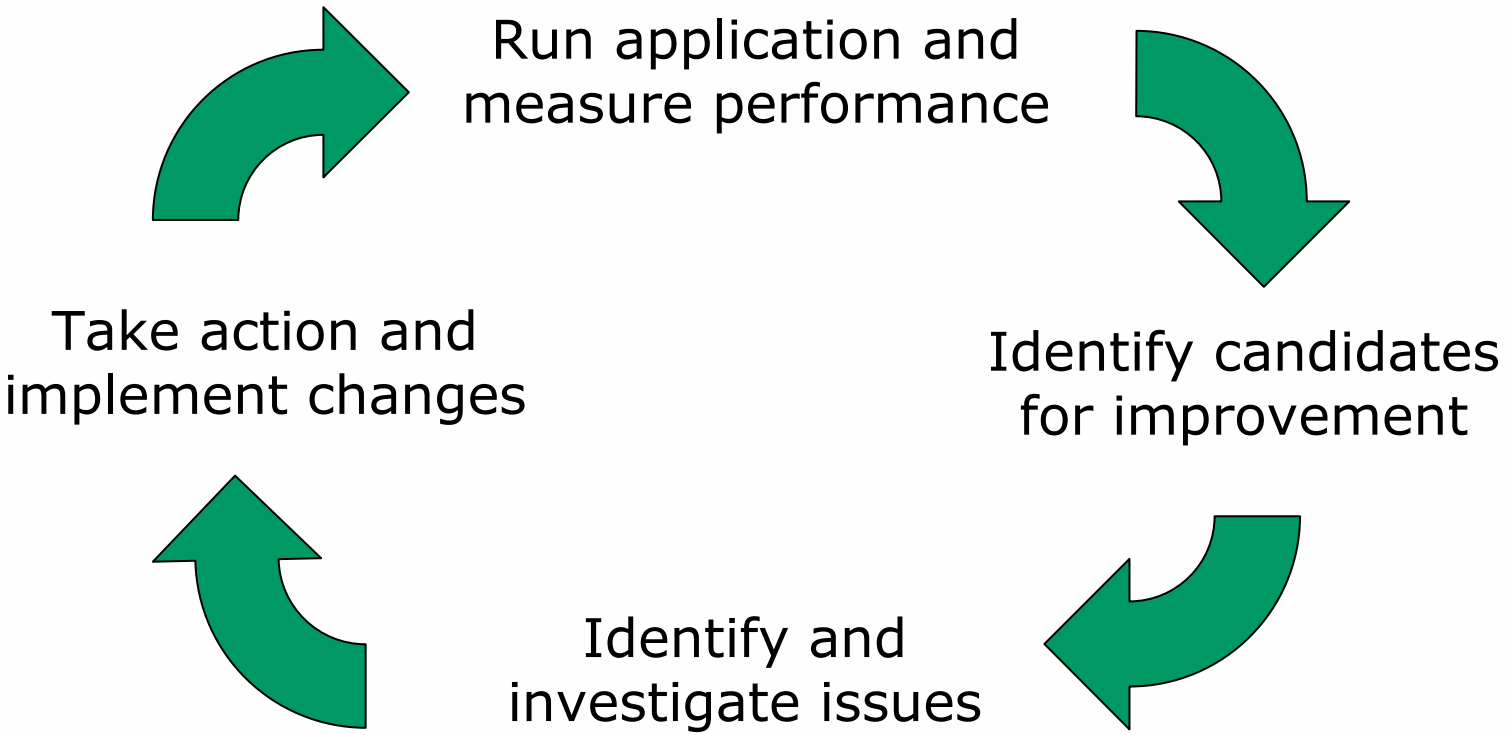
Overview & Optimisation Session

Overview

AMD
Smarter Choice

- Overview of the performance Tuning Process
- Performance measurement
- CodeAnalyst for Windows
- CodeAnalyst for Linux
- Performance analysis tips and pitfalls
- Features under investigation/development
- Q & A

Performance tuning process



Performance Tuning Basic Rules

AMD
Smarter Choice

- Keep performance concept in the product development cycle
- Develop reliable code before performance tuning
- Focus on the algorithm first
- Take the advantage of compiler options and hardware processor features

Performance Measurement

AMD Smarter Choice

- Measure the execution time of application
- Traditionally use RDTSC to measure elapsed time on code block

```
XOR EAX, EAX
CPUID
RDTSC
MOV [clock], EAX
//
//Your Code Block
//
XOR EAX, EAX
CPUID
RDTSC
SUB EAX, [clock]
```

Performance Measurement – Cont.

AMD Smarter Choice

- Each code has its own timestamp counter in a multi-core system
- Timestamp counters are not guaranteed to be in sync. Therefore, comparing two values from RDTSC can be dangerous.
- OS move process among cores
- Use:

```
QueryPerformanceCounter()
QueryPerformanceFrequency()
GetTickCount()
```

CodeAnalyst™ for Windows

AMD
Smarter Choice

- **Major features:**

- **Timer-based profiling (TBP)**
 - On APIC enabled Windows systems, the finest time resolution is 0.1ms and 1.0ms non-APIC enabled systems.

- **Event-based profiling (EBP)**
 - Designed to profile the 32 public performance events of AMD Athlon, and all 104 performance events on AMD Opteron and AMD Athlon 64 and event combination.
 - With event multiplexing technique, CodeAnalyst could profile more than 4 events per run on AMD hardware.
 - Event counts are interpreted in meaningful manner, like IPC, cache miss rate etc.

CodeAnalyst™ for Windows – Cont.

AMD
Smarter Choice

- **Call Stack Sampling**
 Unwind the call stack to collect call path data on hotspots and display caller/callee relationship on hotspots during postprocessing.

- **Thread Analysis**
 Identify thread numbers, thread affinity, thread/CPU activity and remote memory access.

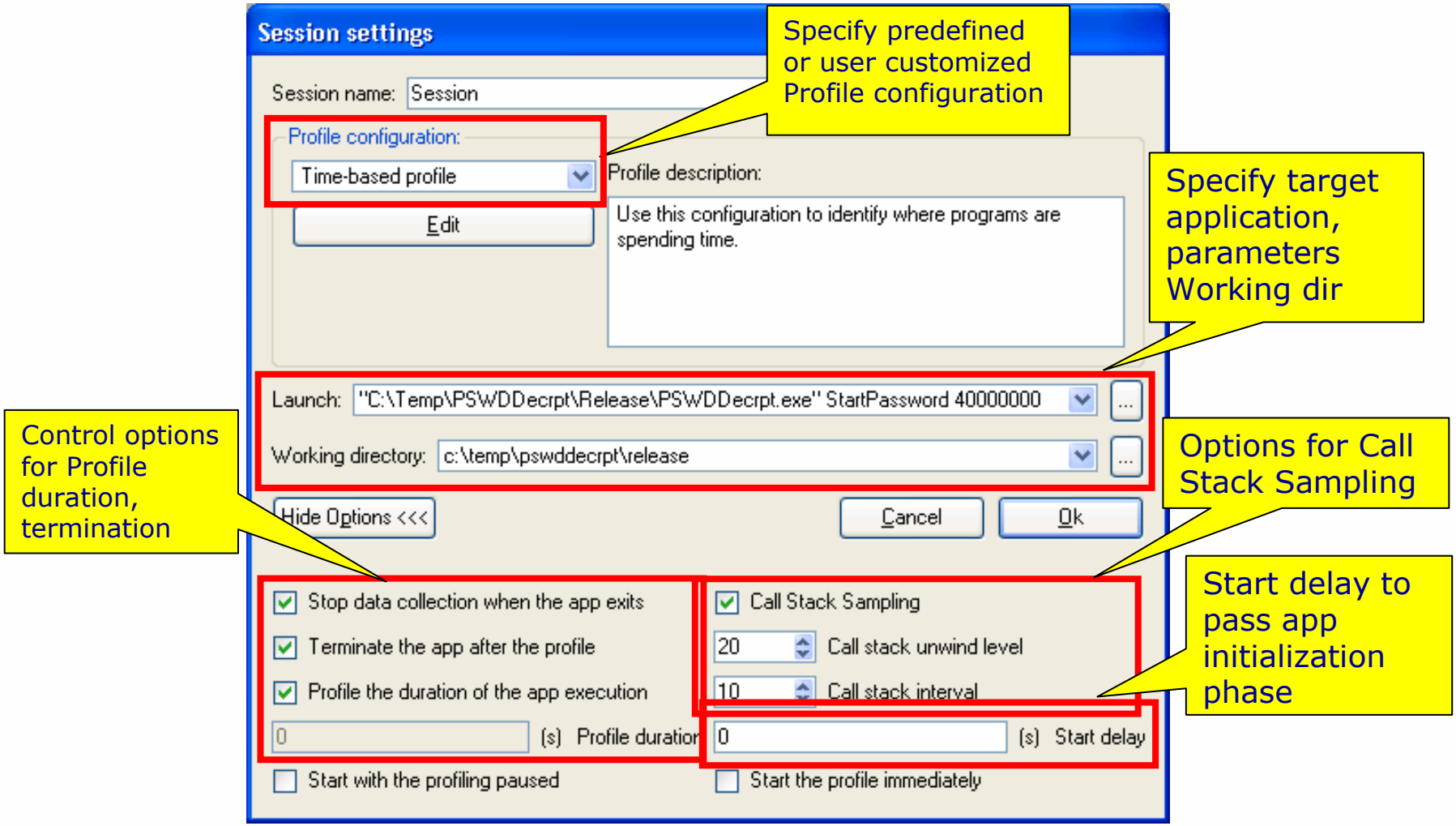
- **Trace generation and Pipeline Simulation**
 Graphical display of execution pipeline events with penalty and dependency annotations (for Windows environment only at this time)

CodeAnalyst™ for Windows – Cont. 2

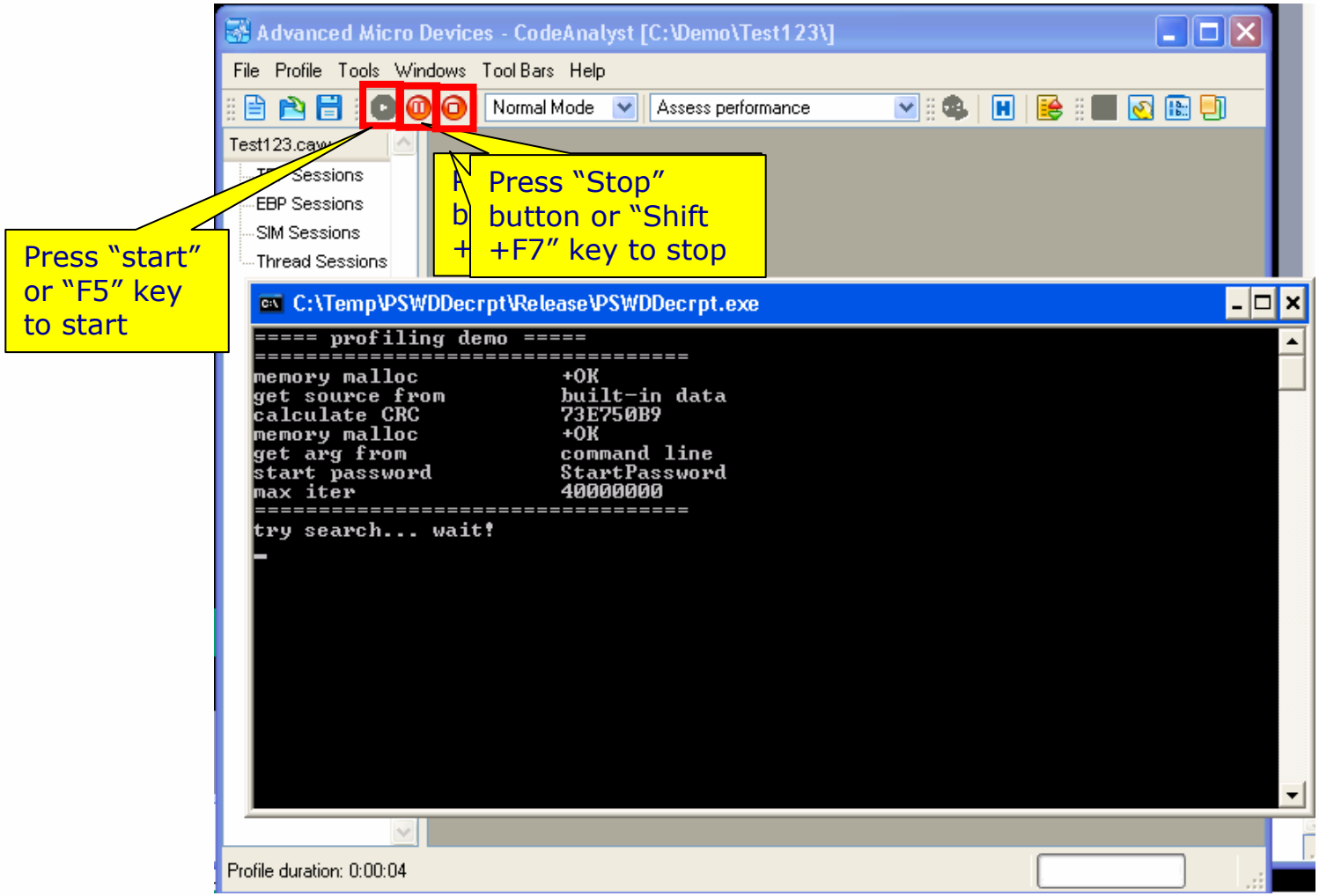
AMD Smarter Choice

- **Other features:**
- **Low overhead, system-wide profiling**
 Can profile dynamically loaded modules
 Can profile kernel mode drivers as well as applications
- **Supports multiprocessor profiling**
- **Supported Microsoft Operating Systems:**
 Windows NT, Windows 2000
 32/64 bit Windows XP, Windows Server 2003
 32/64 bit Windows Vista, Longhorn
- **GUI to ease performance analysis**
- **Command line utilities**
 Supports execution scripts
- **Supports profiling of JIT code**
 Java, Microsoft .NET, Driver JIT code
- **Plug-in Profile integrated into Visual Studio 2005**
- **Profile control API library**
 Pause and resume profiling

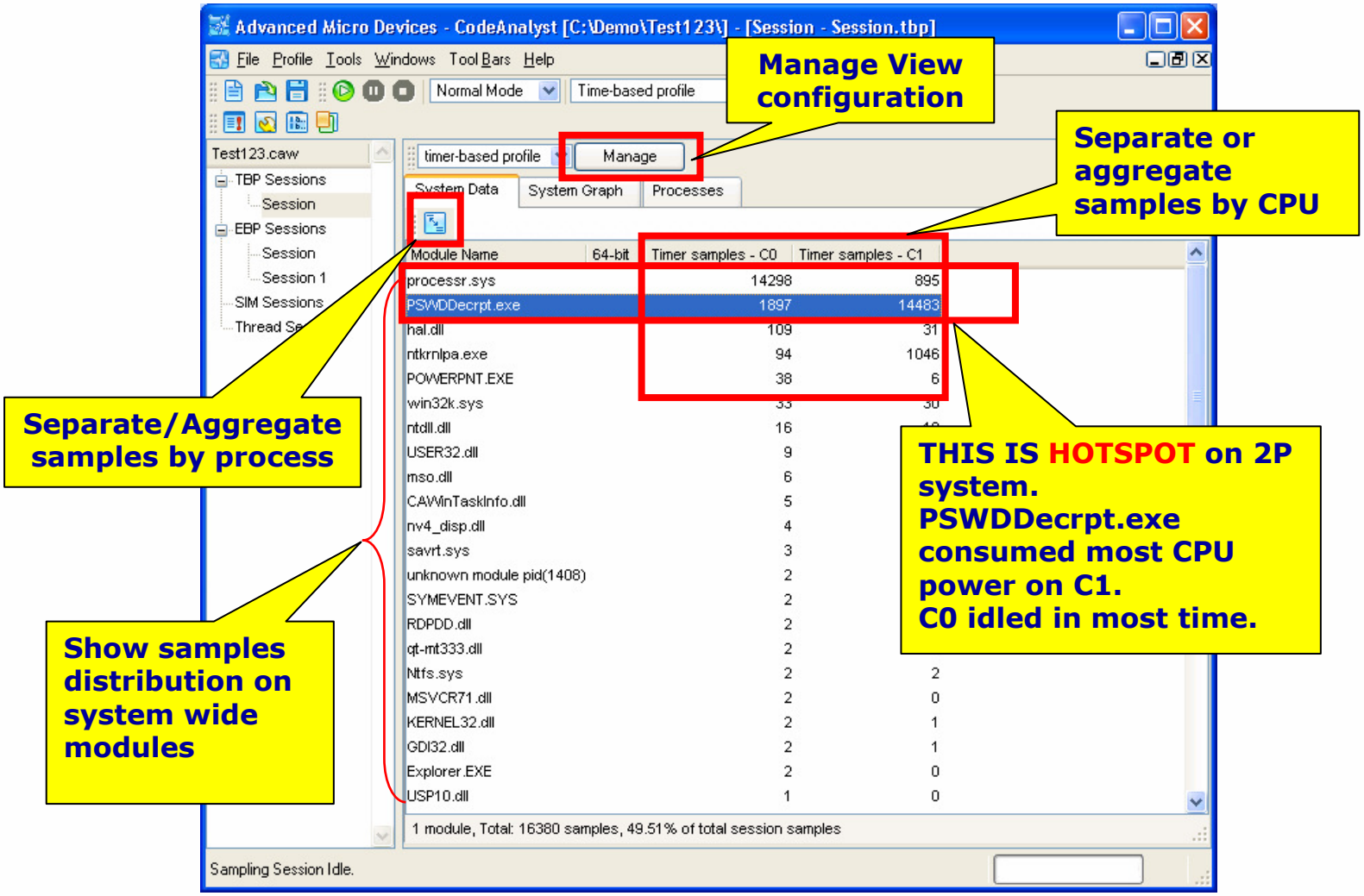
CodeAnalyst™ - TBP Configuration



CodeAnalyst™ - profile in process



CodeAnalyst™ - System Data View

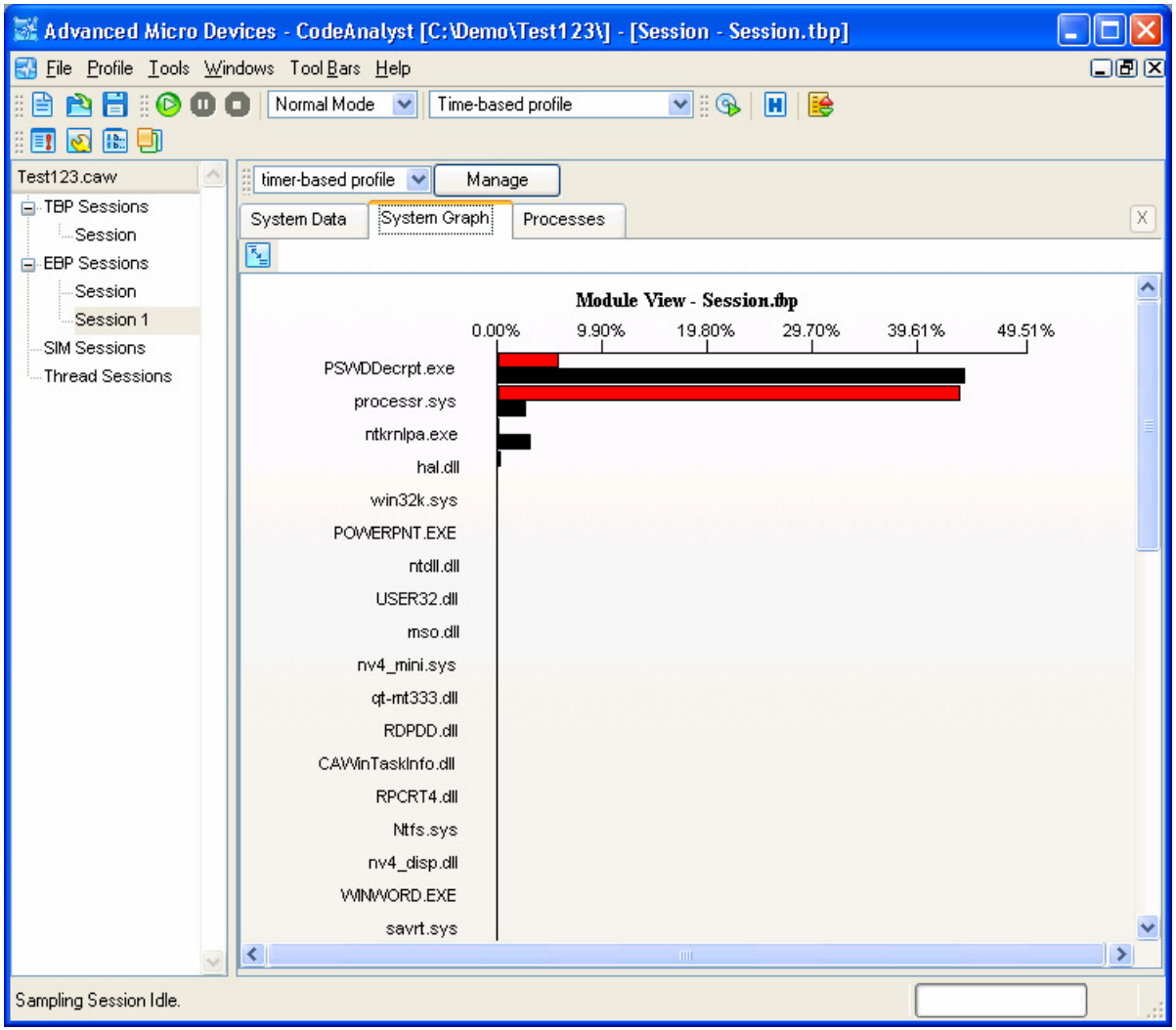


Program Hotspot

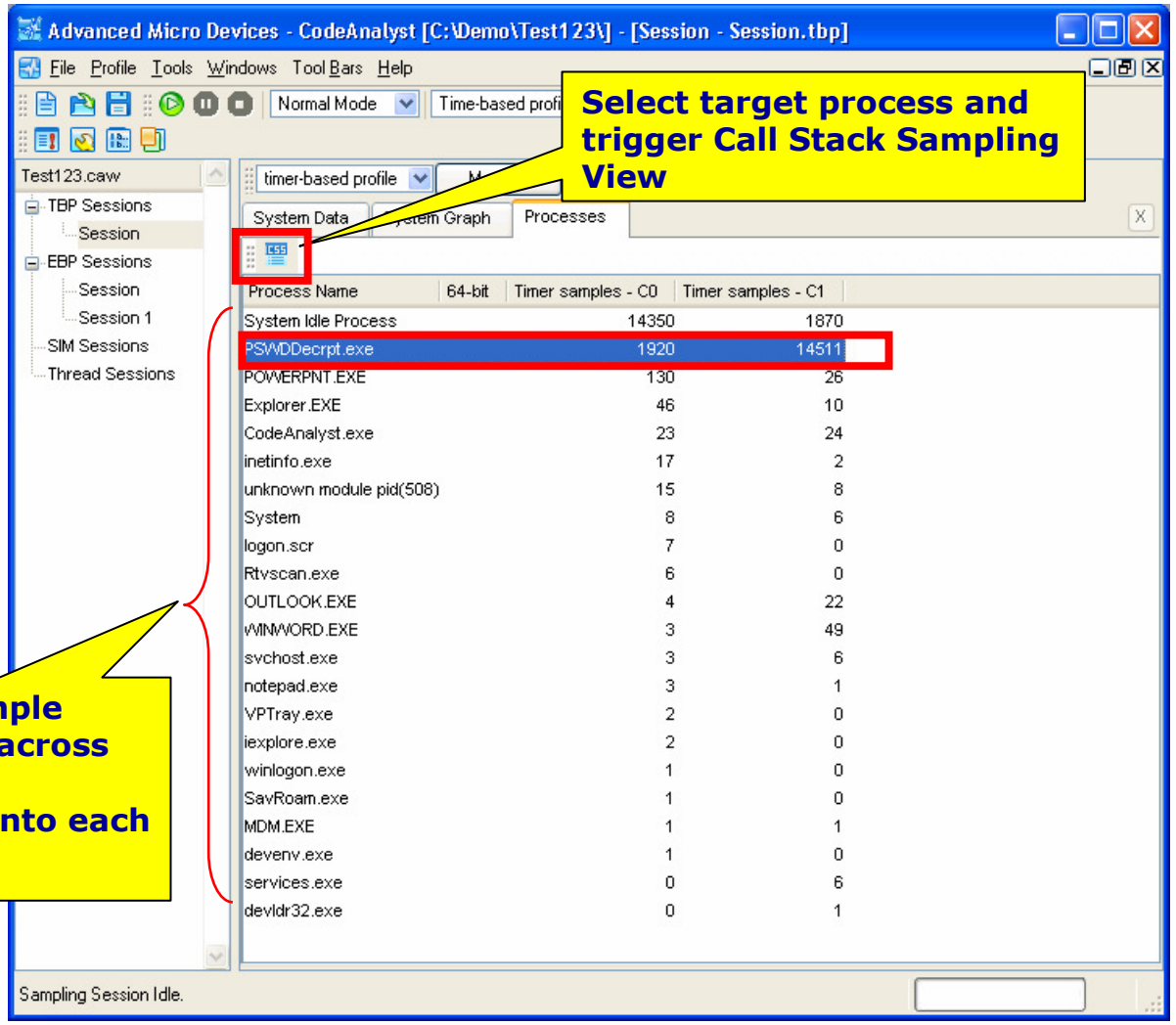
AMD
Smarter Choice

- Hotspots are the areas of the application that have intense activity. It's the place to start optimizing.
- Intense activity usually refers to time, but the cause of a hotspot could be branches mis-prediction, cache misses, etc.
- CodeAnalyst time-based profile helps to identify program hotspot
- CodeAnalyst event-based profile helps to analyze cause of hotspot

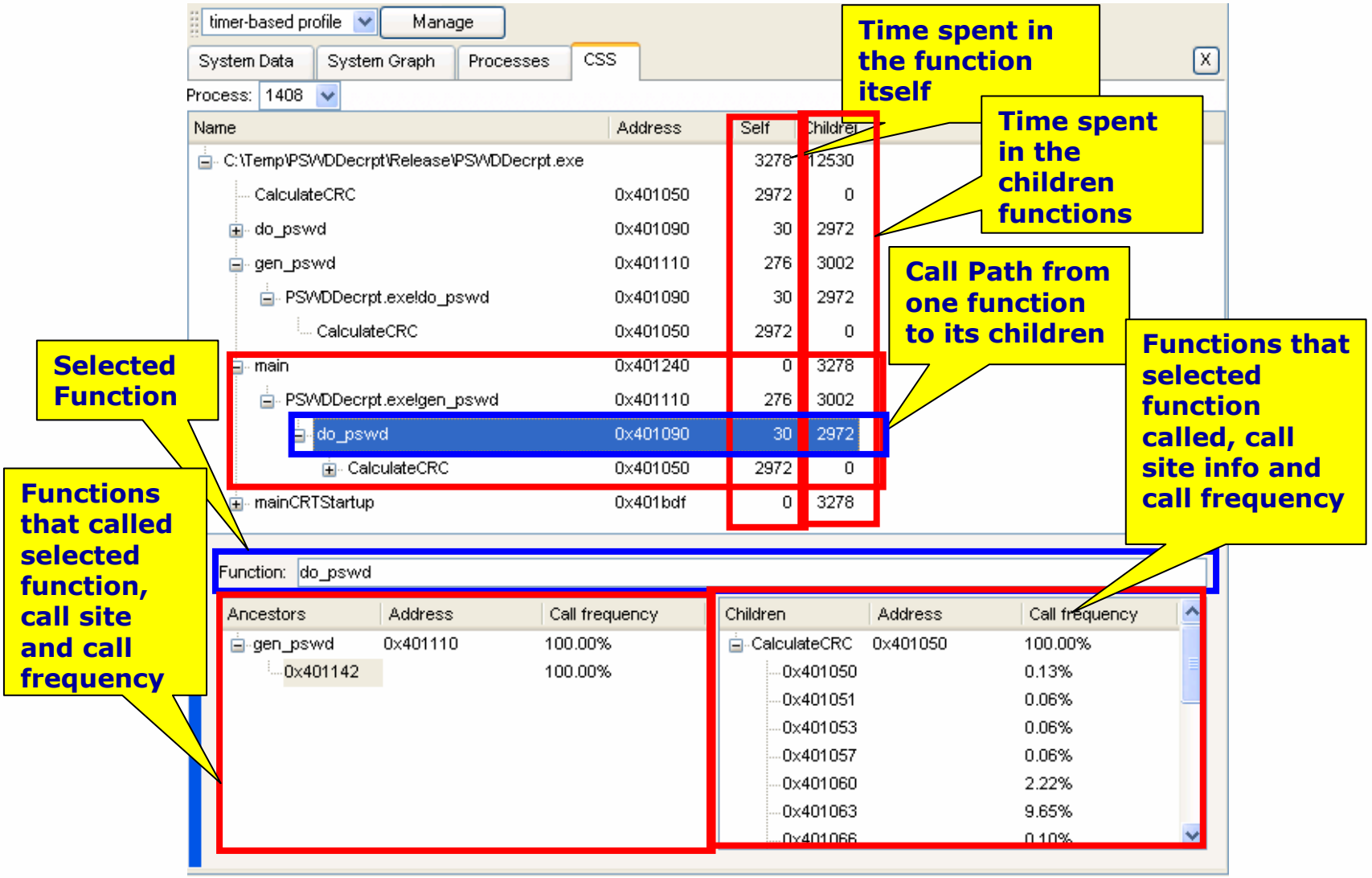
CodeAnalyst™ - System Graph View



CodeAnalyst™ - Process View



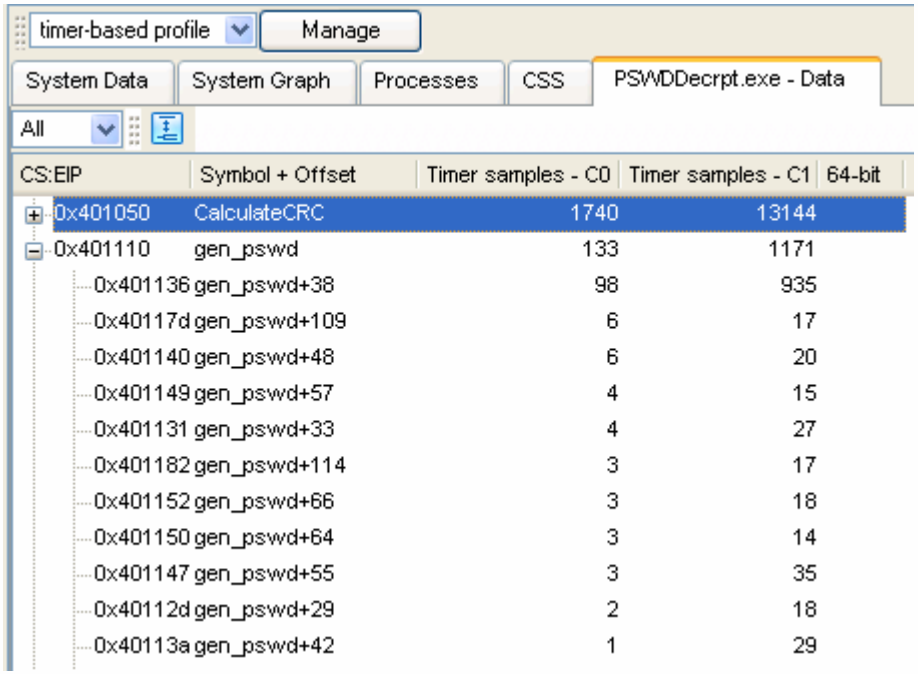
CodeAnalyst™ - Call Path View



CodeAnalyst™ - Module Data View

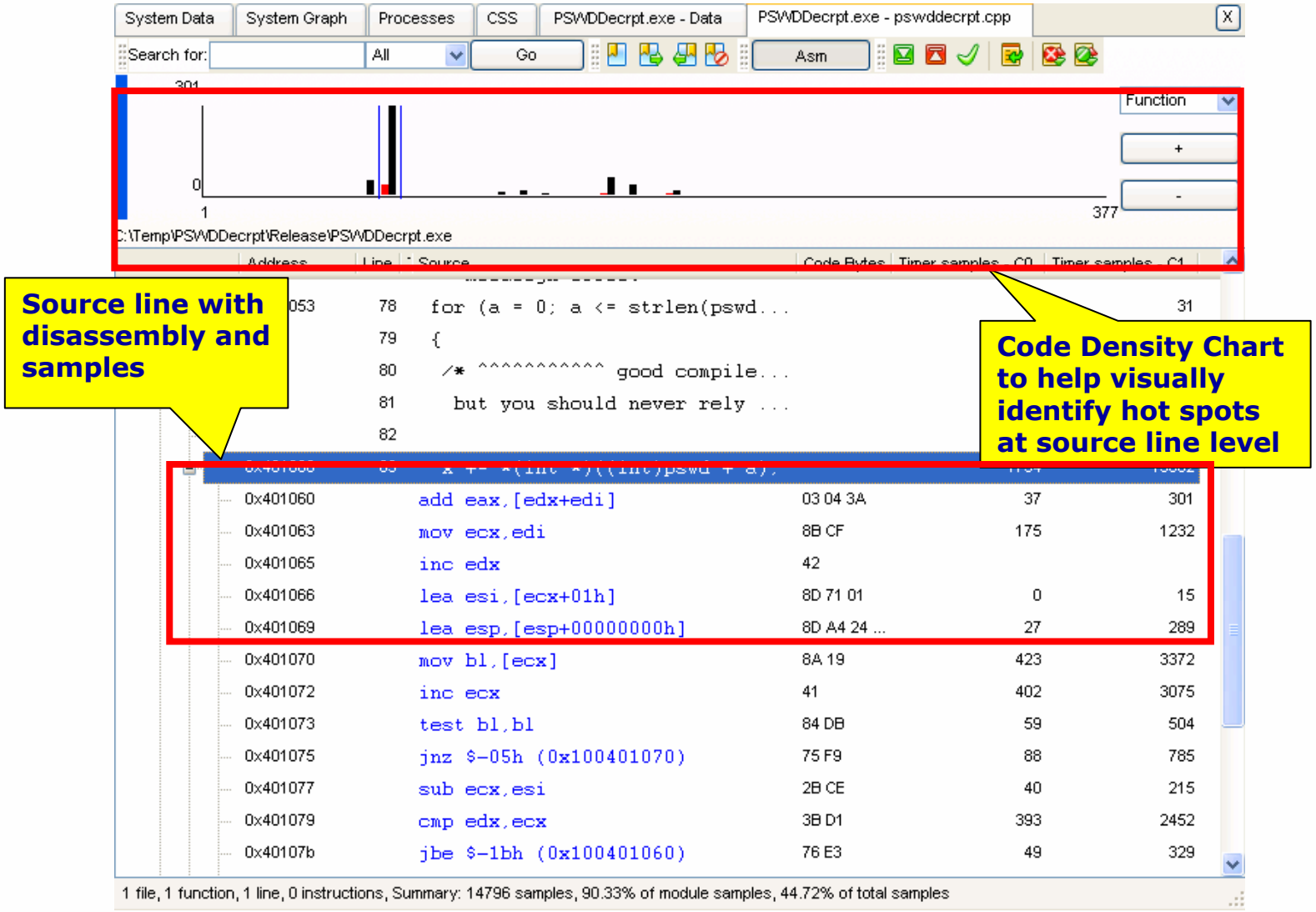
AMD Smarter Choice

- Drill down to functions within a module
- Display a function-by-function time breakdown
- Identify candidate functions for improvement



CS:EIP	Symbol + Offset	Timer samples - C0	Timer samples - C1	64-bit
0x401050	CalculateCRC	1740	13144	
0x401110	gen_pswd	133	1171	
0x401136	gen_pswd+38	98	935	
0x40117d	gen_pswd+109	6	17	
0x401140	gen_pswd+48	6	20	
0x401149	gen_pswd+57	4	15	
0x401131	gen_pswd+33	4	27	
0x401182	gen_pswd+114	3	17	
0x401152	gen_pswd+66	3	18	
0x401150	gen_pswd+64	3	14	
0x401147	gen_pswd+55	3	35	
0x40112d	gen_pswd+29	2	18	
0x40113a	gen_pswd+42	1	29	

CodeAnalyst™ - Source Code View



CodeAnalyst™ System Data View (EBP)

AMD Smarter Choice

CPU column indicates time hotspot

Instruction Per Cycle indicates efficiency

Cache Memory behavior

Data Misalignment

Branch Mis-prediction

System Data | em Graph | Proces

Module Name	CPU clocks	IPC	DC miss rate	DTLB L1M L2M rate	Misalign rate	Mispredict rate
PSWDDecrpt.exe	31717	1.24	0.00		0.01	0.02
ntkrnlpa.exe	2673	1.03	0.00	0.0	0.00	0.00
win32k.sys	317	0.77	0.01	0.0	0.00	0.01
hal.dll	273	0.23	0.06		0	0.00
POWERPNT.EXE	81	0.42	0.04	0.0	0	0.01
ntdll.dll	62	0.47	0.00	0.0	0	0.01
RDPDD.dll	46	0.72	0.00		0	0.01
USER32.dll	44	0.34	0	0.0	0	0.01
Ntfs.sys	27	0.44	0		0.01	0.01
qt-mt333.dll	24	0.25	0		0	0.03
mso.dll	19	0.21	0.15	0.0	0	0
CAWinTaskInfo.dll	19	0.95	0		0	0.02
kernel32.dll	18	0.61	0.01		0	0.04
msmapi32.dll	16	0.88	0		0	0.02
USP10.dll	15	0.60	0.01		0	0.02
nv4_mini.sys	14	0.71	0.03	0.0	0	0
nv4_disp.dll	12	0.33	0.03		0	0

Identify Memory Issues

AMD
Smarter Choice

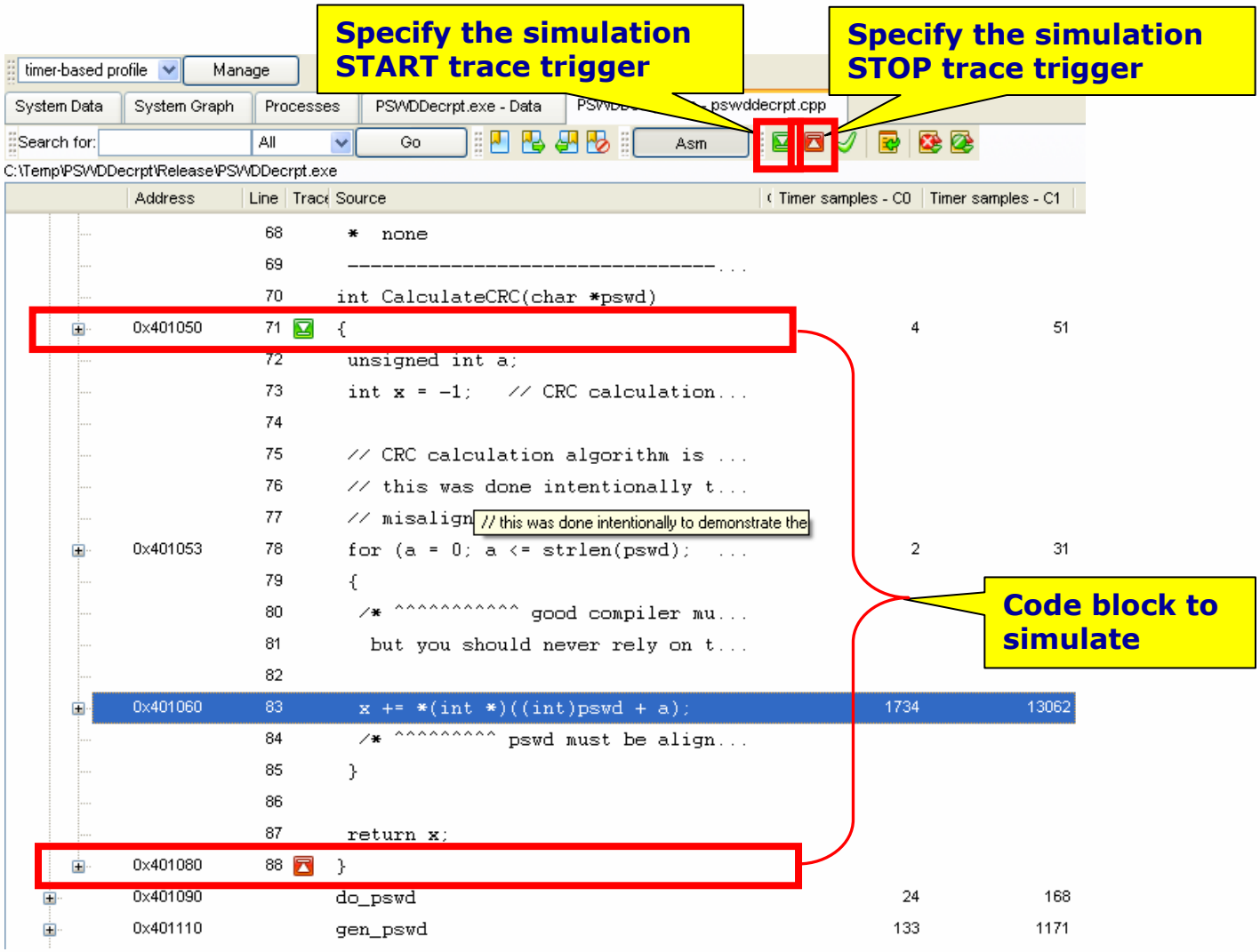
- First, run time-based profile in CodeAnalyst
- Second, run event based profile with 0x41 DCache miss and 0x81 ICache miss.
- If hotspot of time-based profile and event-based profile (Cache miss) point to same place (Function, code block), it is a strong sign that memory accesses are causing the bottleneck.
- If hotspot of TBP and the address of cache miss are not correlated, the Optimisation efforts should focus on hotspot of TBP first.

CodeAnalyst™ - Pipeline Simulation

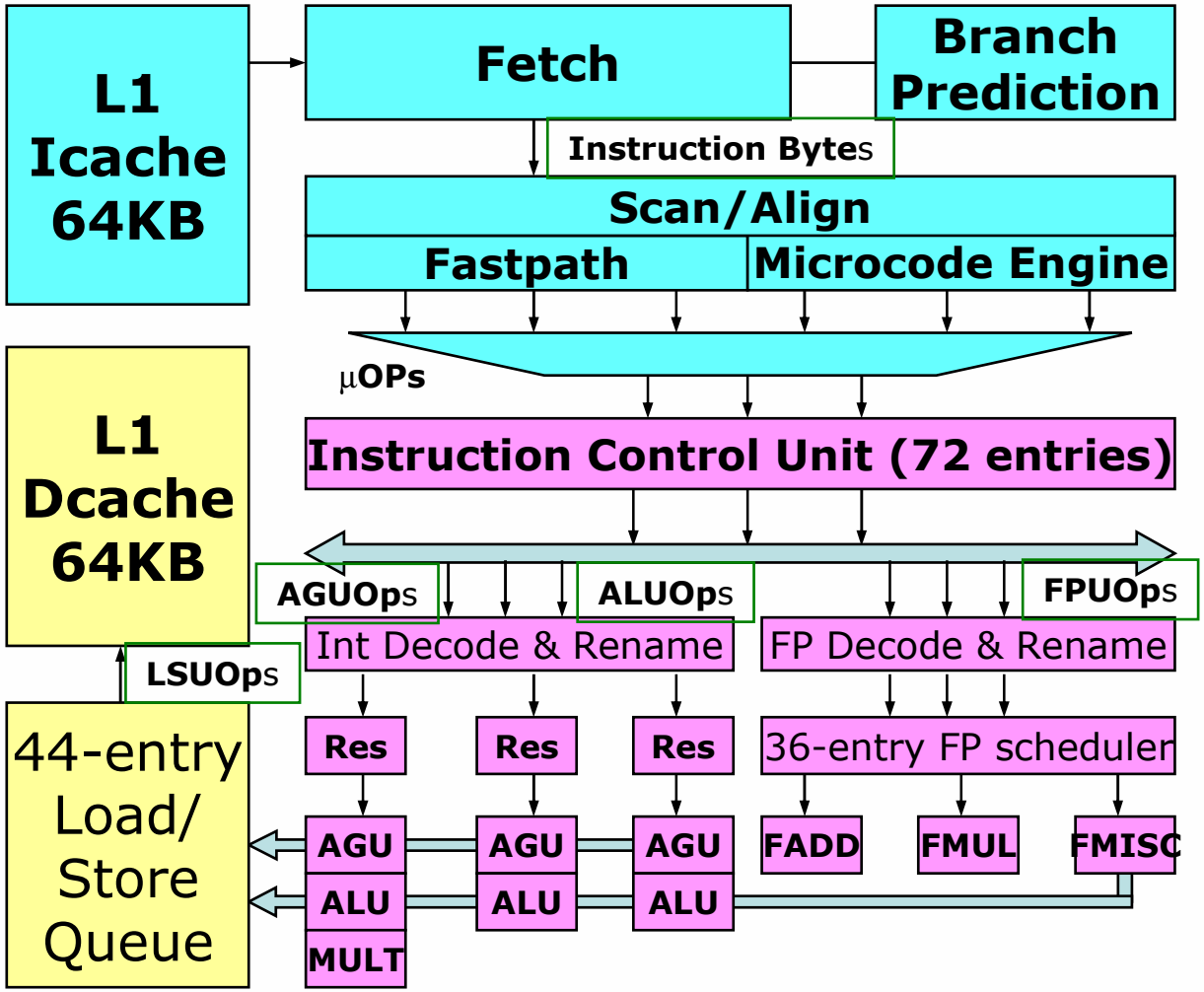
AMD Smarter Choice

- CodeAnalyst can simulate a user-specified code region on an AMD microprocessors and obtain cycle-precise execution info, such as stalls, dependencies, and penalty, etc.
- The procedure of CodeAnalyst pipeline simulation:
- Users define a block of code, -- by specifying “start” and “stop” trace point.
- CodeAnalyst launches the target application.
- Once “Start” trace point is hit, CodeAnalyst will collect runtime info, like instruction, register, TLB etc, and write it into a trace file.
- Feed trace into simulator.
- Process simulation results and provide graphic pipeline display

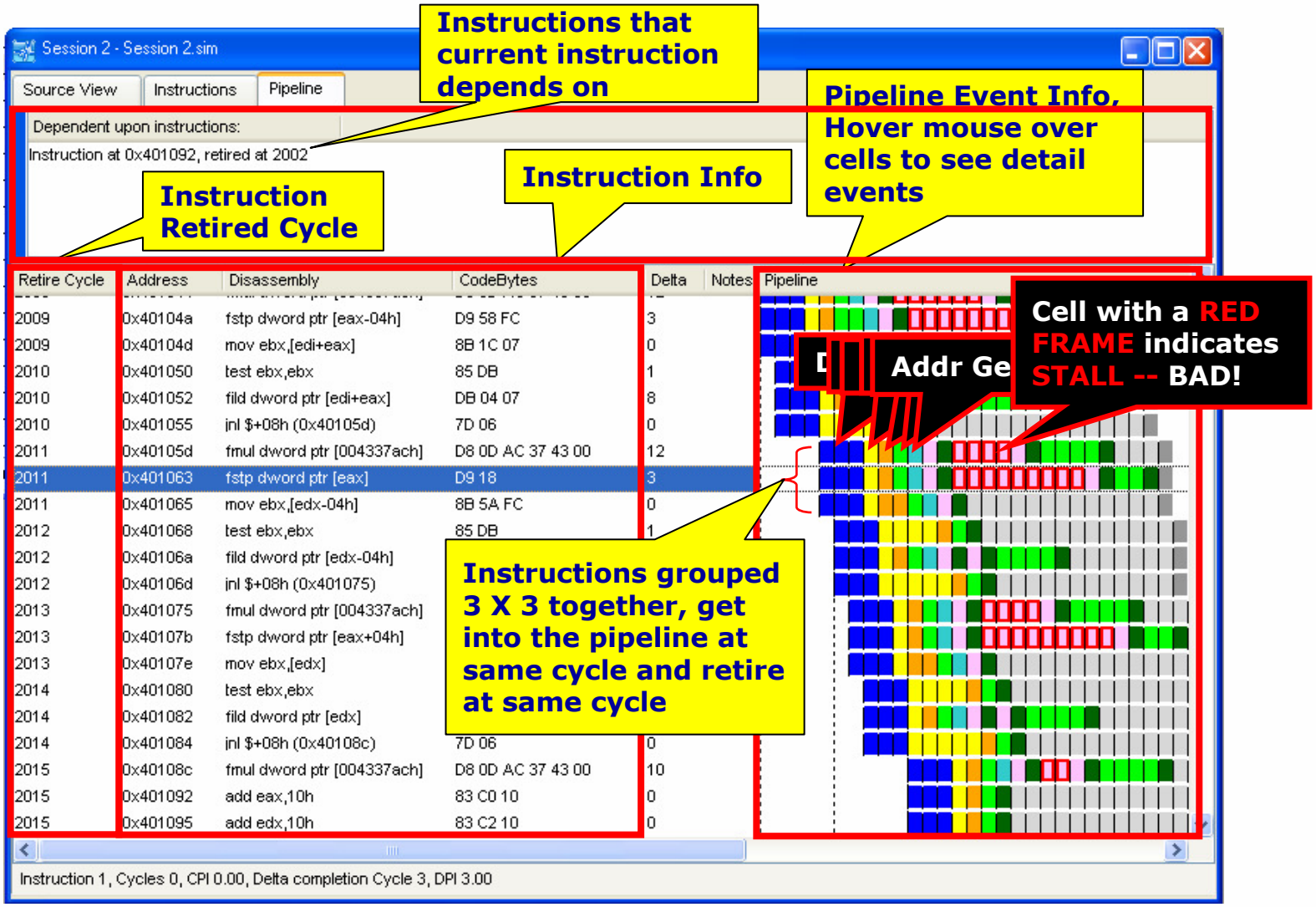
CodeAnalyst™ - Simulation Setup



Simplified Processor Core



CodeAnalyst™ - Pipeline View

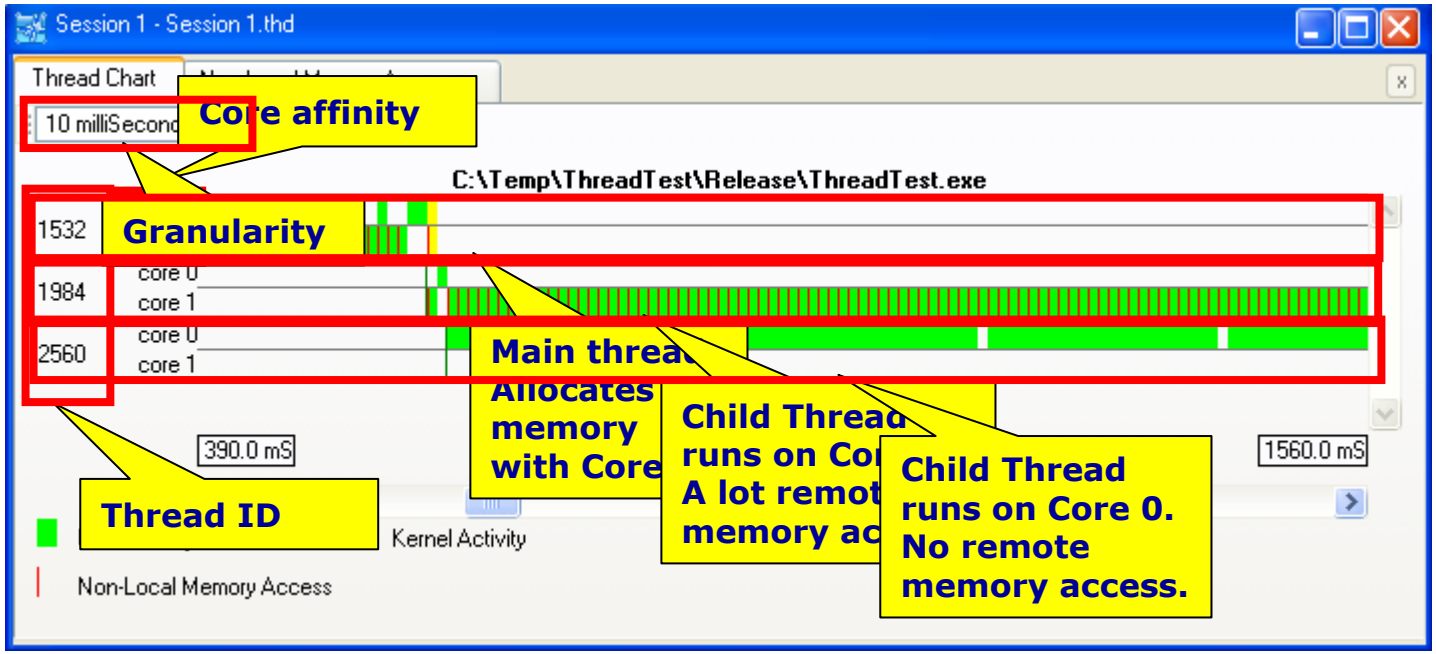


Penalties

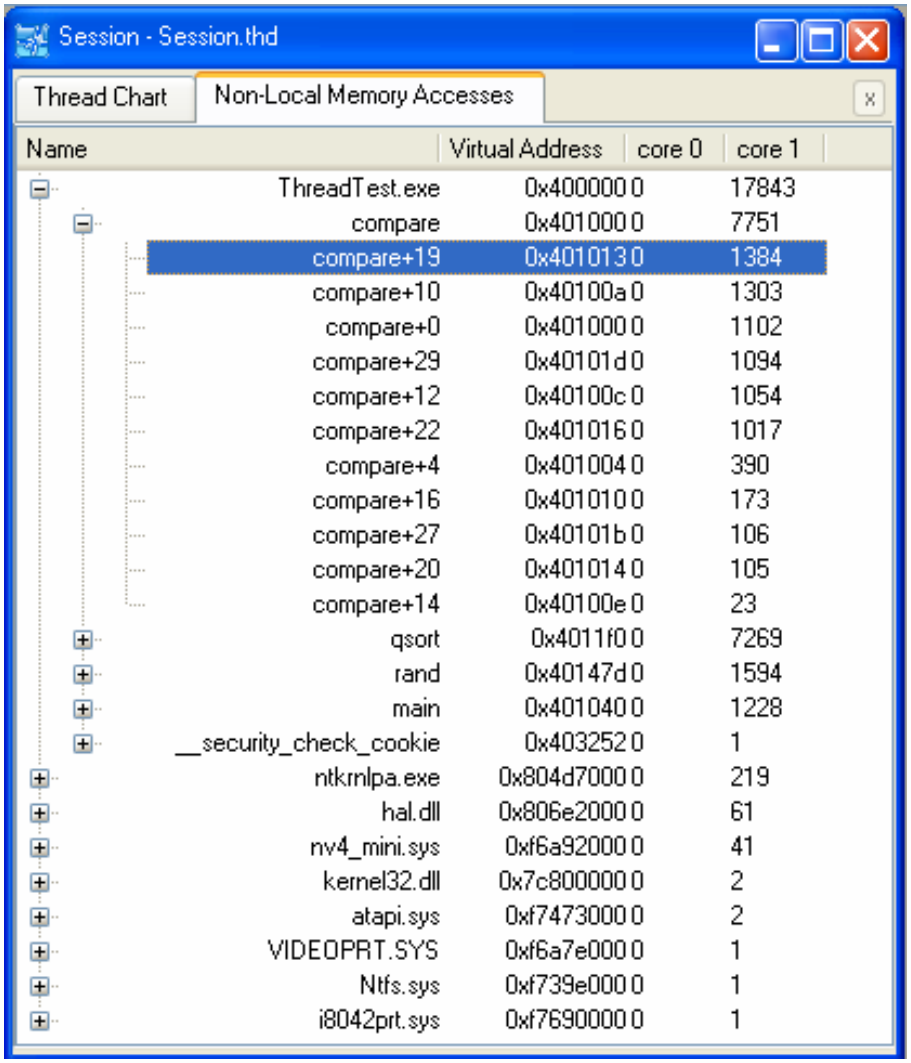
Bottleneck Causes Execution Penalties Notes Annotations	Description	Penalty
Bank conflict	Reference bank conflicted	1+ cycles
Br misprdct	Branch was mispredicted	10+ cycles
BT split wdw	Branch target splits fetch windows	1+ cycles
IC miss	Instruction miss in instruction cache	10+ cycles
C3 misprdct	Jump to 0xc3 opcode caused mispredict	10+ cycles
DC miss	Reference missed in DC	9+ cycle
DTLB miss L0	Reference missed in Data Table L0	2+ cycles
DTLB miss L1	Reference missed in Data Table L1	10+ cycles
DW misalign	Dword reference is misaligned	1+ cycle
ITLB miss L0	Instruction fetch missed in L0 Instruction Table	2+ cycles
ITLB miss L1	Instruction fetch missed in L1 Instruction Table	10+ cycles
Ld/St Q full	Load/Store queue is full	No fixed penalty
No FCOMP	FCOMP does **not** precede FSTSW	Causes code to execute sequentially
No predecode	Instruction was not predecode	7+ cycles
Partial write	Write to a partial register	Causes code to execute sequentially
QW misalign	Qword reference is misaligned	1+ cycle
Serialize inst	Serializing instruction	Huge penalty in terms of cycles. Avoid this penalty.
Stlf 2:0	Store->load forward 11:3 match 2:0 mismatch	Causes code to execute sequentially
Stlf 31:12	Store->load forward 31:12 mismatch	Causes code to execute sequentially
Stlf Ld Misaligned	Store->load forward load is misaligned	Causes code to execute sequentially
Stlf opsize	Store->load forward opsize violation	Causes code to execute sequentially
Stlf St Misaligned	Store->load forward store is misaligned	Causes code to execute sequentially
WC flush	Write combining buffer flushed	No fixed penalty
Wd misalign	Word reference is misaligned	1+ cycle

CodeAnalyst™ - Thread View

- Uses sampling to measure activity
- Shows user/kernel activity, thread Creation Termination, thread core affinities



Non-Local Memory Accesses

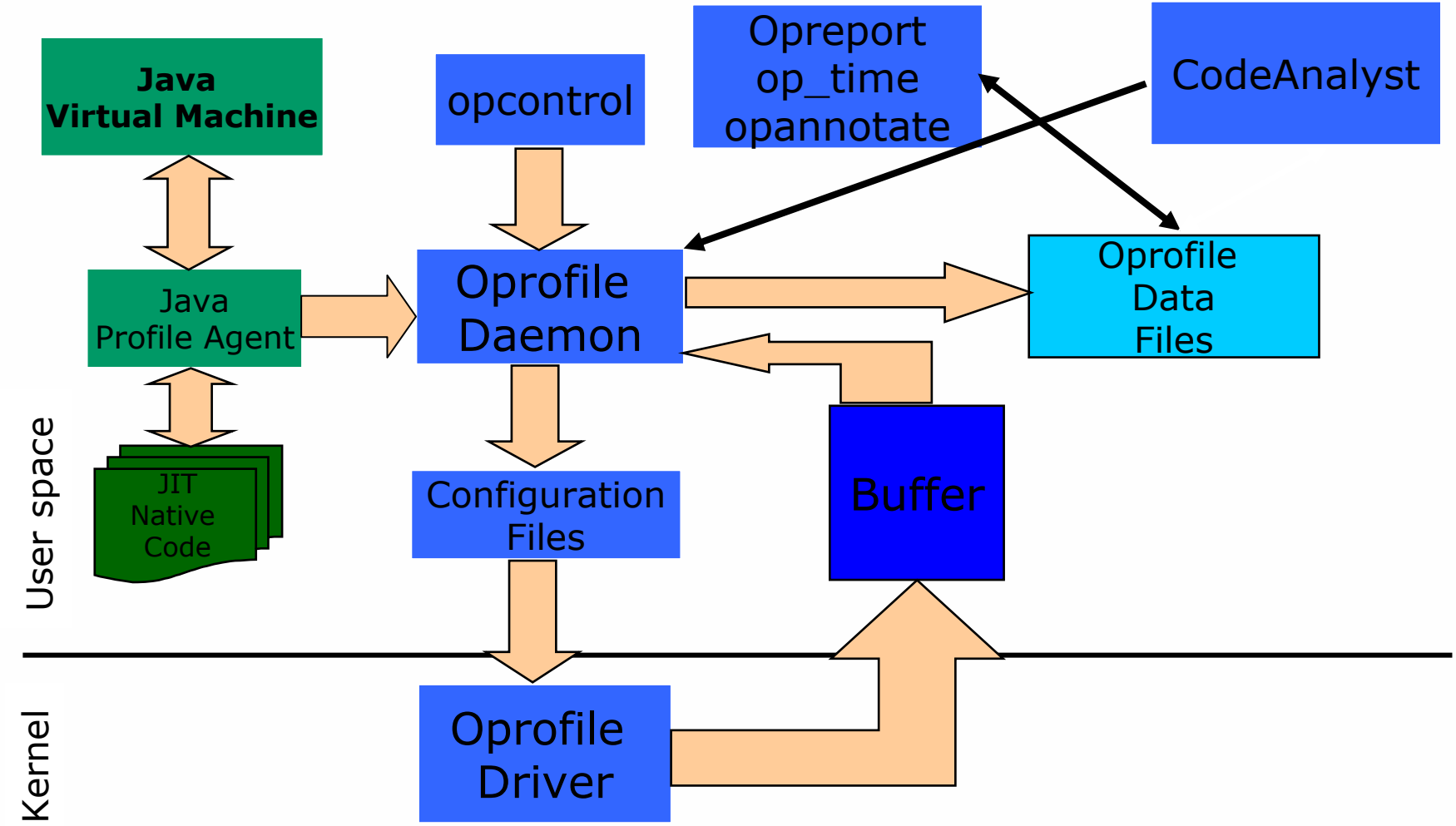


CodeAnalyst™ for Linux

AMD
Smarter Choice

- Built on top of the Oprofile driver and daemon
- CodeAnalsyt 2.5 modified driver and daemon to support Java-profiling
- CodeAnalsyt 2.6 modified daemon only for Java profiling. Work on SLES 10, RHEL 5 etc.
- Purpose is to provide ease of profile configuration and quicker data analysis
- Consistent user interface with CodeAnalyst Windows for cross-platform ease of use
- Similar profile data drill down and graphical display as provided in CodeAnalyst Windows
- Trace and pipeline simulation are provided as command line utilities in simulation tool kits.

Oprofile/CodeAnalyst™ Architecture



CodeAnalyst™ Tips and Workaround

AMD Smarter Choice

- Need input/output redirection?
 - Use input/output redirection in batch and launch batch file in CodeAnalyst
- Microsoft symbol server support
 - Microsoft symbol server provides symbol file for system modules.
 - Warning: It could slow down the post process.
- Run target app on 1 core of multi-core system
 - Change CPU flag in boot.ini
 - Set “processor affinity” of cmd.exe, launch target from command line prompt.
- Simulation import

CodeAnalyst™ Pitfalls and Limitations

AMD Smarter Choice

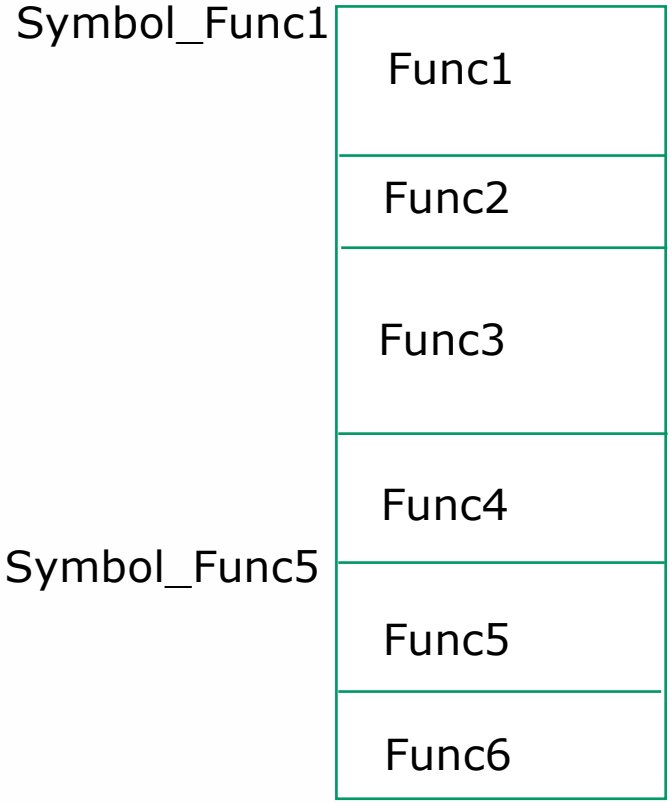
- The address associated with a sample does not always indicate the instruction that caused an event because:

- Due to out-of-order execution, it's hard to identify which instruction caused interrupt.
- Athlon/Athlon64 pack up to 3 instructions into pipeline simultaneously.
- CodeAnalyst is using the return address on stack of interrupt service routine.

Function and Symbol pitfall

AMD
Smarter Choice

- Without debug info, CodeAnalyst cannot obtain source info.
- Some functions do not have symbols associated with them.
- Module code section is separated by available symbols.
- Profile samples in Func1-Func4 will be counted under symbol_Func1.



CodeAnalyst™
Features under Investigation

AMD
Smarter Choice

- New sampling mechanism which will gather precise sample RIP information.
- Thread analysis:
 - Monitor thread activities and identify synchronization issue
 - Detect data race and Thread API violation etc
- Call Graph Profiling:
 - Complete call path
 - Graphic representation

CodeAnalyst™ Download and Contact

AMD Smarter Choice

- Free!
- Download from AMD Developer Central
 http://www.amd.com/codeanalyst/
- Send feedback, bug report, feature request to:
 support.codeanalyst@amd.com

Optimisation Session Outline

AMD
Smarter Choice

- Exploiting Performance Opportunities
 - Obvious Performance Potential
 - Hidden Performance Potential
 - Exposing Untapped Performance Potential
 - Analyzing Performance Improvement Trials

Obvious Performance Potential

- Processor Architecture
 - x64 Processors
 - *Extended Memory Addressing*
 - *Additional Registers*
 - *Deeper Execution Pipeline*
 - Multi-Core Processors
 - *Multiprocessing for the desktop system*
 - Multiple processor platforms
- 64-bit operating systems
- Compiler Optimisation switches
- Optimized libraries (for example AMD ACML)

Hidden Performance Potential

AMD
Smarter Choice

- Efficient algorithms
- Cache friendly memory access
- Branch Prediction friendly conditionals
- Parallel work through Threads
- Object Synchronization

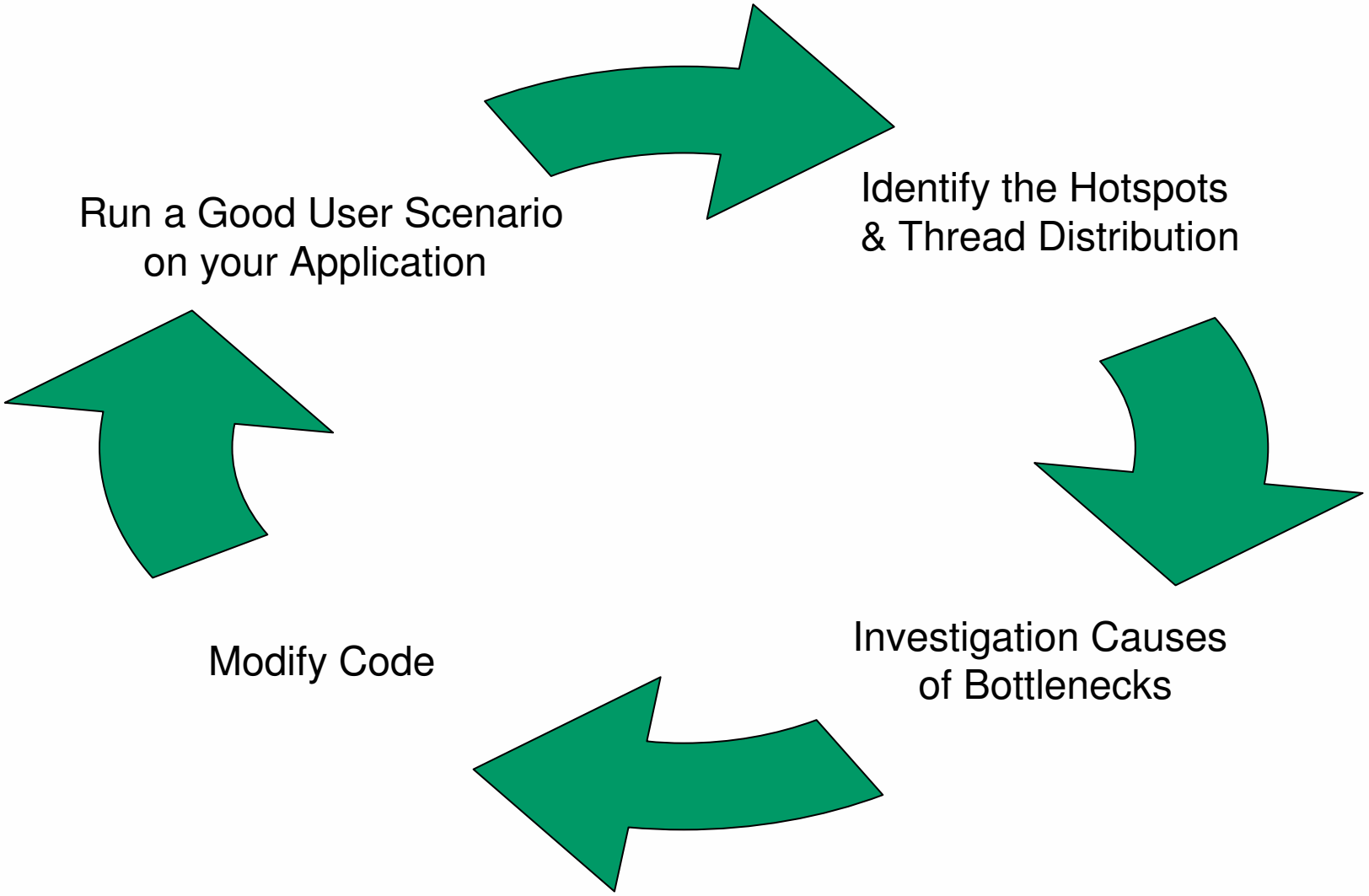
Expose Untapped Performance Potential

AMD
Smarter Choice

- **Timer-based sampling -** Where is the application or system spending its time?
 → Identify time consuming or frequently executed code possibly pointing to algorithm issues (Hot Spots)
- **AMD Opteron™ and AMD Athlon™ 64 processor performance events** - How is the application behaving on the CPU? Memory?
 → Evaluate the applications use of architectural features
- **Thread View** - How are threads behaving on cores? Remote/local access?
 → Evaluate effective use of multiple processors.
- **Pipeline Simulation -** Where are delays occurring in the pipeline?

 → Understand how data dependencies can stall the processor execution

Analyzing Performance



Timer-based Profiling
1st Level of Analysis

AMD
Smarter Choice

- Exposes areas of intense activity
- Identifies the most likely suspects
- Provides a sample distribution chart
- Ability to drill down through several data views
- View the source code on and around the sample
- Algorithmic issues may be evident from the hot spot code
- Hot spot code might suggest particular events to profile in next level of Analysis

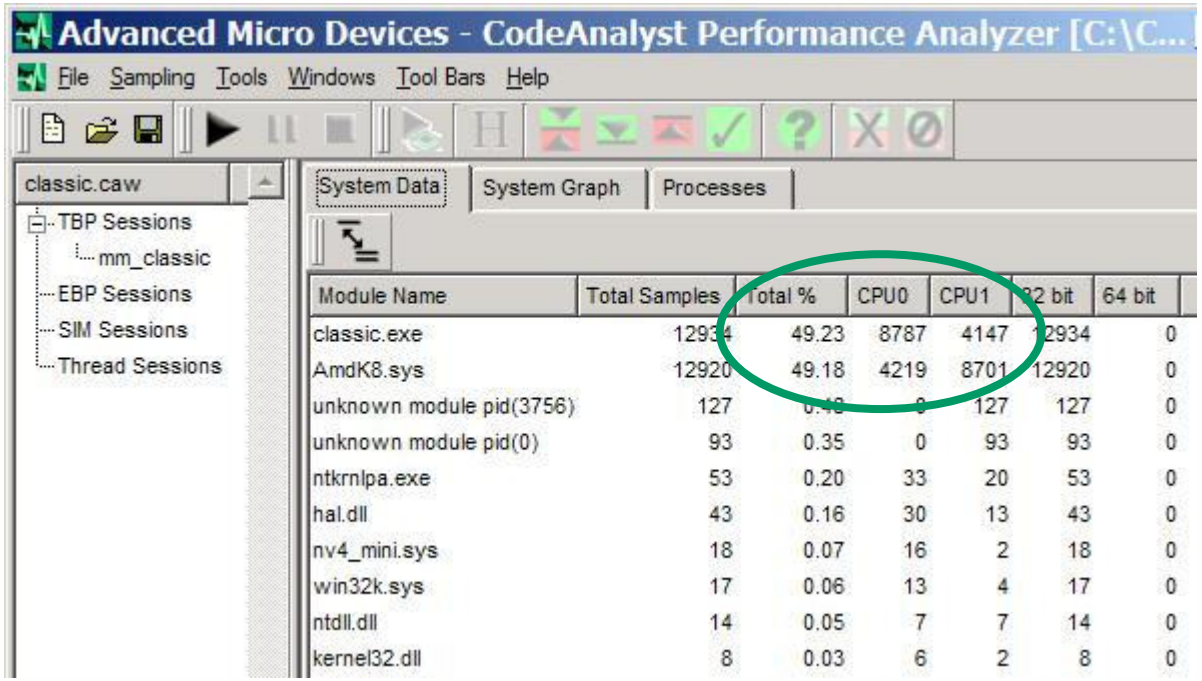
Common Hot Spots

AMD
Smarter Choice

- Loops
 - Large content and large loop counts are natural hot spots but not bad for performance
 - Small content with small fixed loop counts should be unrolled
 - Remove redundant constant calculations from inner loops, including from inner control structures
- Long Logical Expressions in If Statements
- Long data dependent expressions
- Complicated Floating Point expressions

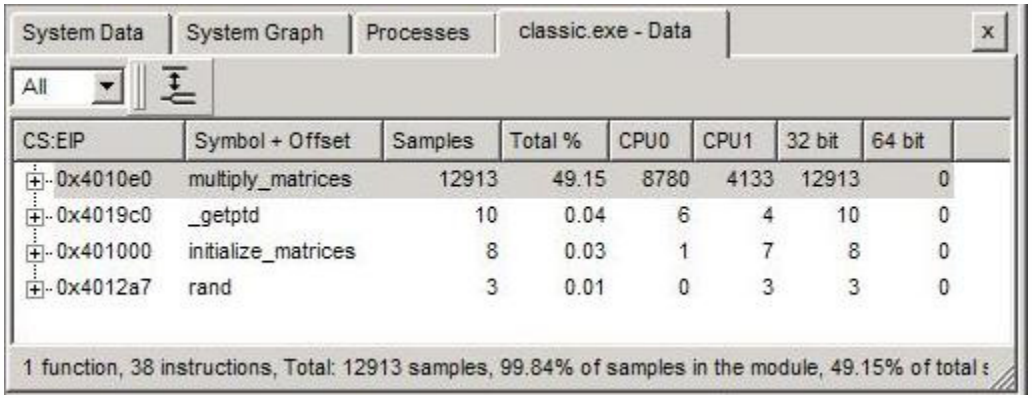
Timer-based profiling

- Display time by module or by process
- Drill down into module or process for more detail
- Shows time on each core



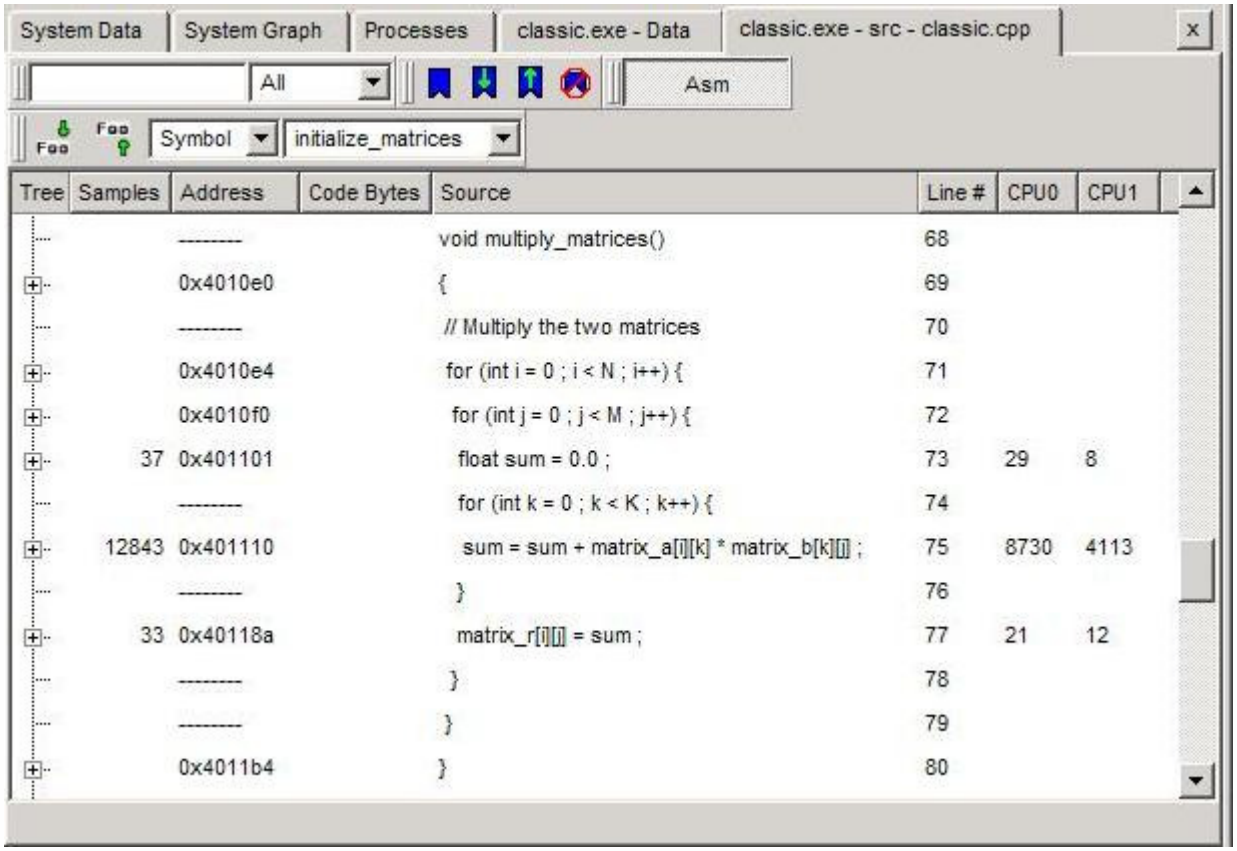
Timer-based profiling

- Drill down to functions within a module
- Display a function-by-function time breakdown
- Identify candidate functions for improvement



Timer-based profiling

- Drill down to source code or even instructions
- Identify code to investigate and tune



Event-based Profiling
2nd Level of Analysis

AMD
Smarter Choice

- Execute Events
- Floating Point Unit Events
- Load/Store Unit and TLB Events
- Instruction Cache Events
- Data Cache Events
- L2 Cache and System Interface Events
- Memory Controller Events
- Crossbar and HyperTransport™ Interface Events
- Total of 78 Events for AMD Opteron™ processor

Example events

- Dispatched FPU ops
- Data cache miss
- CPU clock cycle unhalted
- Retired FPU instructions
- Retired fastpath double op instructions
- Dispatch stall for serialization
- Memory controller bypass counter saturation
- L1 ITLB miss and L2 ITLB hit
- L1 ITLB miss and L2 ITLB miss
- L1 DTLB miss and L2 DTLB hit
- L1 DTLB miss and L2 DTLB miss
- Misaligned data memory references

Examples of Memory Issues

AMD
Smarter Choice

- Large data structures with variable size members not sorted by size
- Use of pointer notation in manipulating large data arrays
- Dereferenced pointer arguments inside a function
- Large declarations of local variables declared randomly with respect to size
- Memory buffers shared between threads

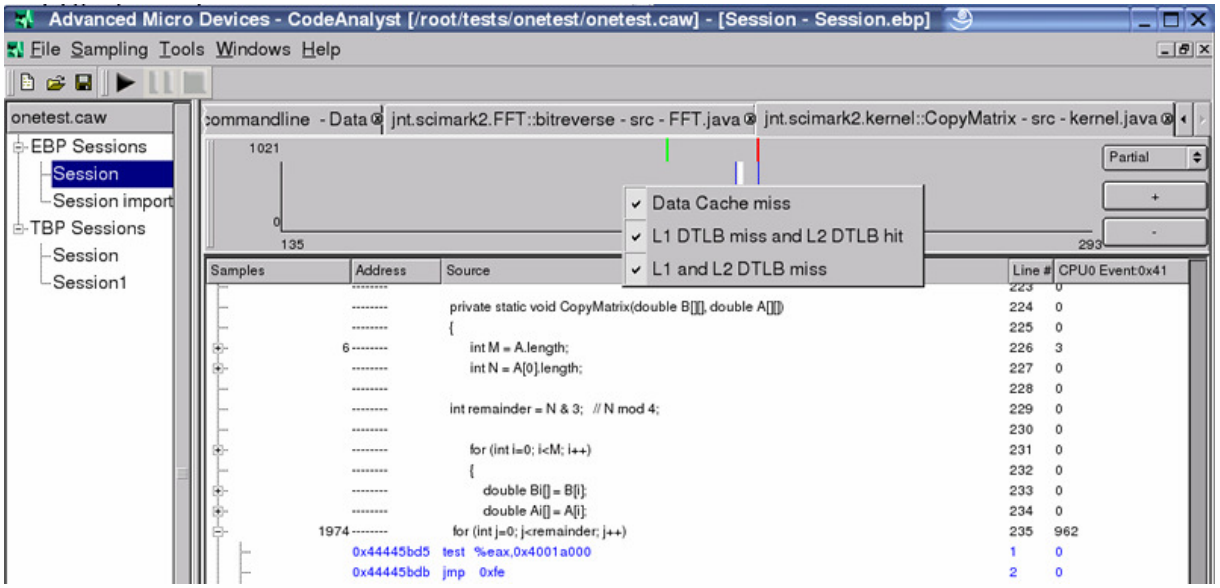
Examples of Branch Prediction Issues

AMD
Smarter Choice

- Order of the expressions in compound branch conditions
- Order of operands in Logical expressions
- Large switch statements with noncontiguous expressions
- Large switch statements cases out of order in respect to probability

Event-based profiling

- Data display and drill down is similar to Timer-basedProfile
- Display event breakdown by module, process, core
- Display event data down to source line or instruction
- High event count and event type helps identify suspect code and issue



Pipeline Simulation Capabilities

AMD
Smarter Choice

- CodeAnalyst can simulate a user specified block of code on AMD microprocessors and provide cycle-precise execution info.

- Requirement:
- Defining a code block to simulate, requires the user to provide debug info for the target module.

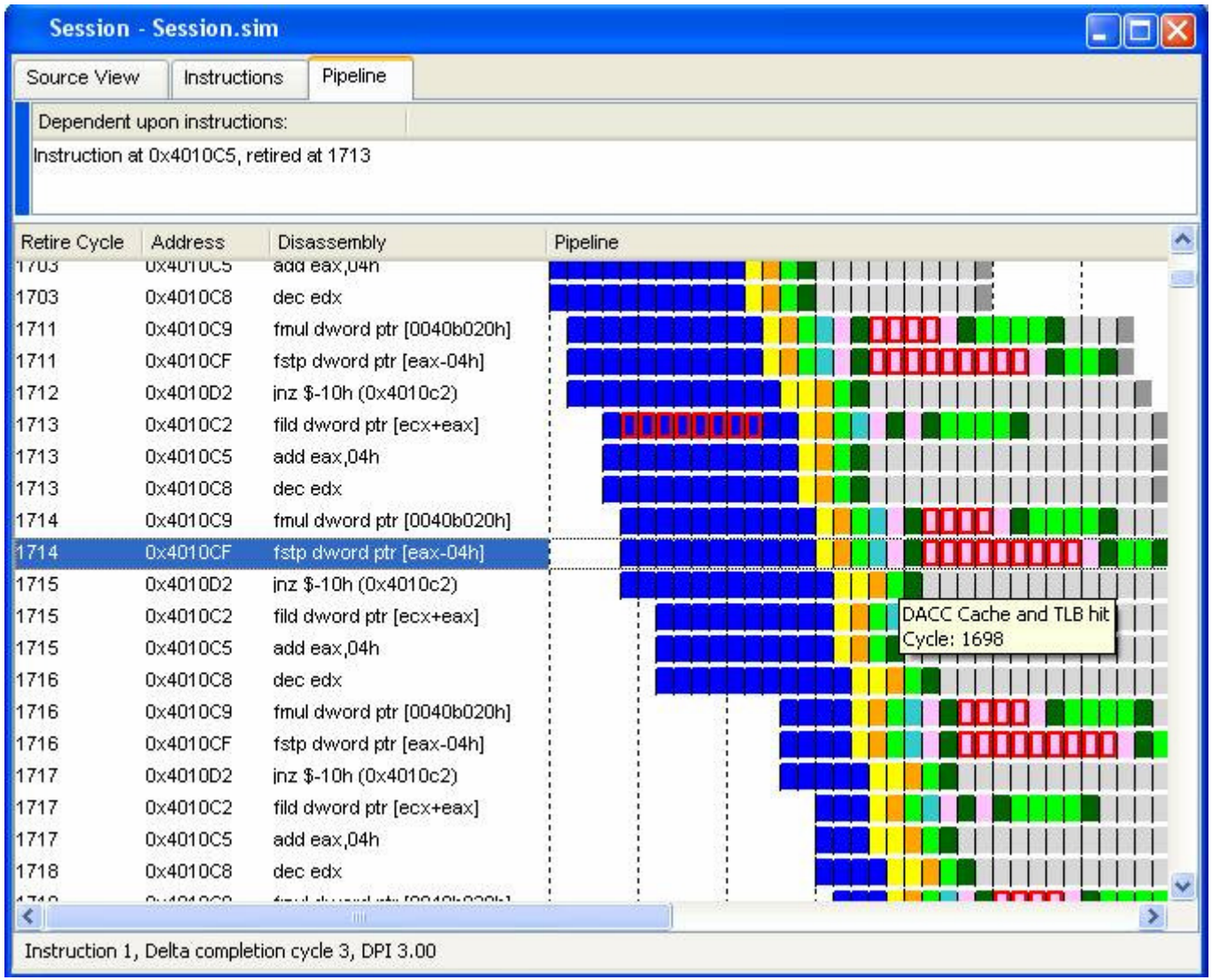
- Limitation:
- Cannot simulate instructions inside system space
- Cannot simulate multi-thread

Some Assumptions in the Simulator

AMD
Smarter Choice

- Assumes perfect memory subsystem
 - All Load/Store Micro-ops hit in the Data Cache
 Assumes that 1 misaligned load = 2 back-to-back aligned loads (64-bit)
 Assumes no cache bank conflicts
 - 100% Instruction cache hit rate
- Assumes perfect branch prediction
- Assumes all schedulers are of infinite size

Pipeline Data View



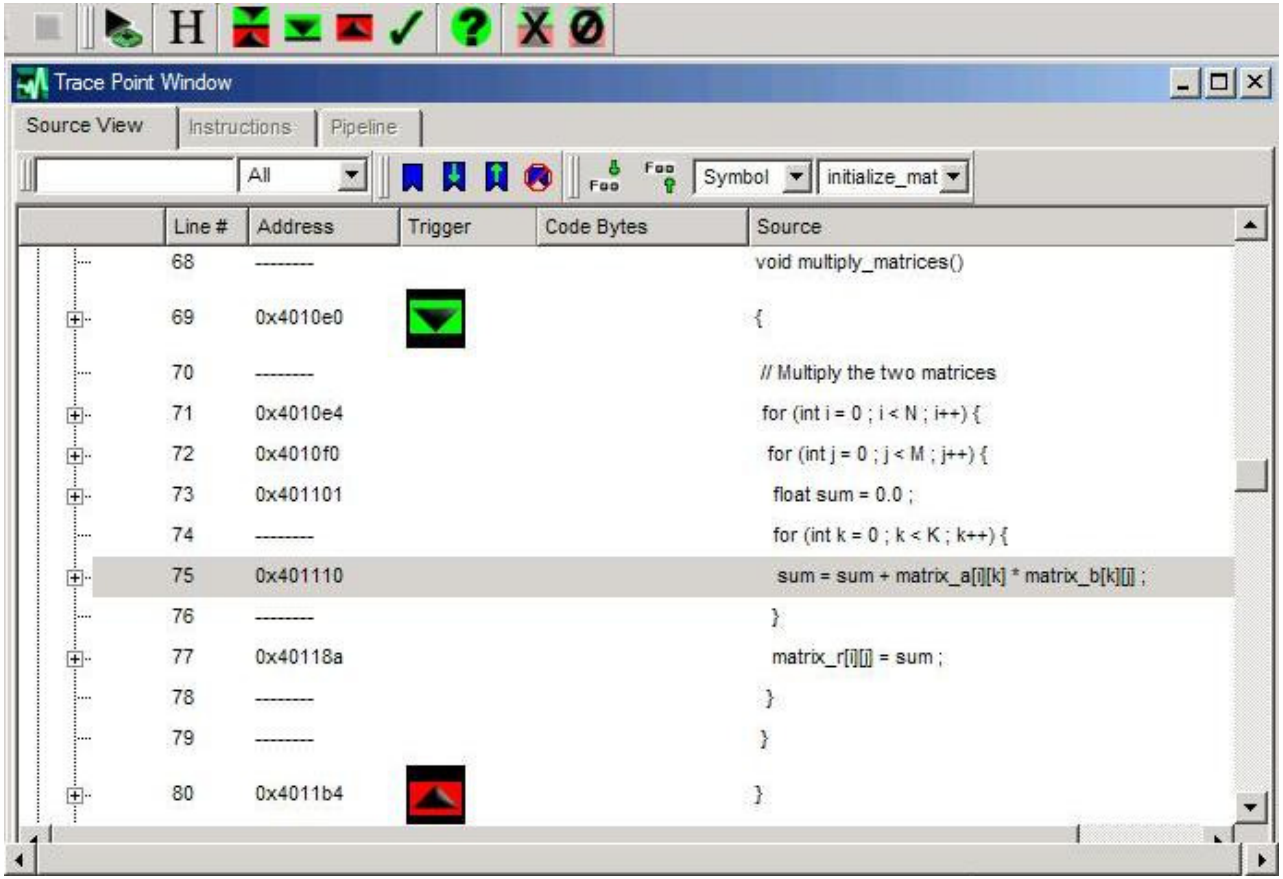
CodeAnalyst Simulation Analysis

AMD
Smarter Choice

- User specifies Simulation configuration
- User sets Trace Point Start, Trace point End, and trace trigger
- Pipeline Data View
 - Pipeline stage
 - Penalty
 - Dependency
 - Delta completion
 - IPC
- User can view Simulation History

Pipeline simulation

- Choose hot spot in code to investigate
- Set start/stop trace points, then click “start” button



Pipeline simulation

- Display cycle-by-cycle instruction execution history
- Hover mouse over cycle to see cause of delay

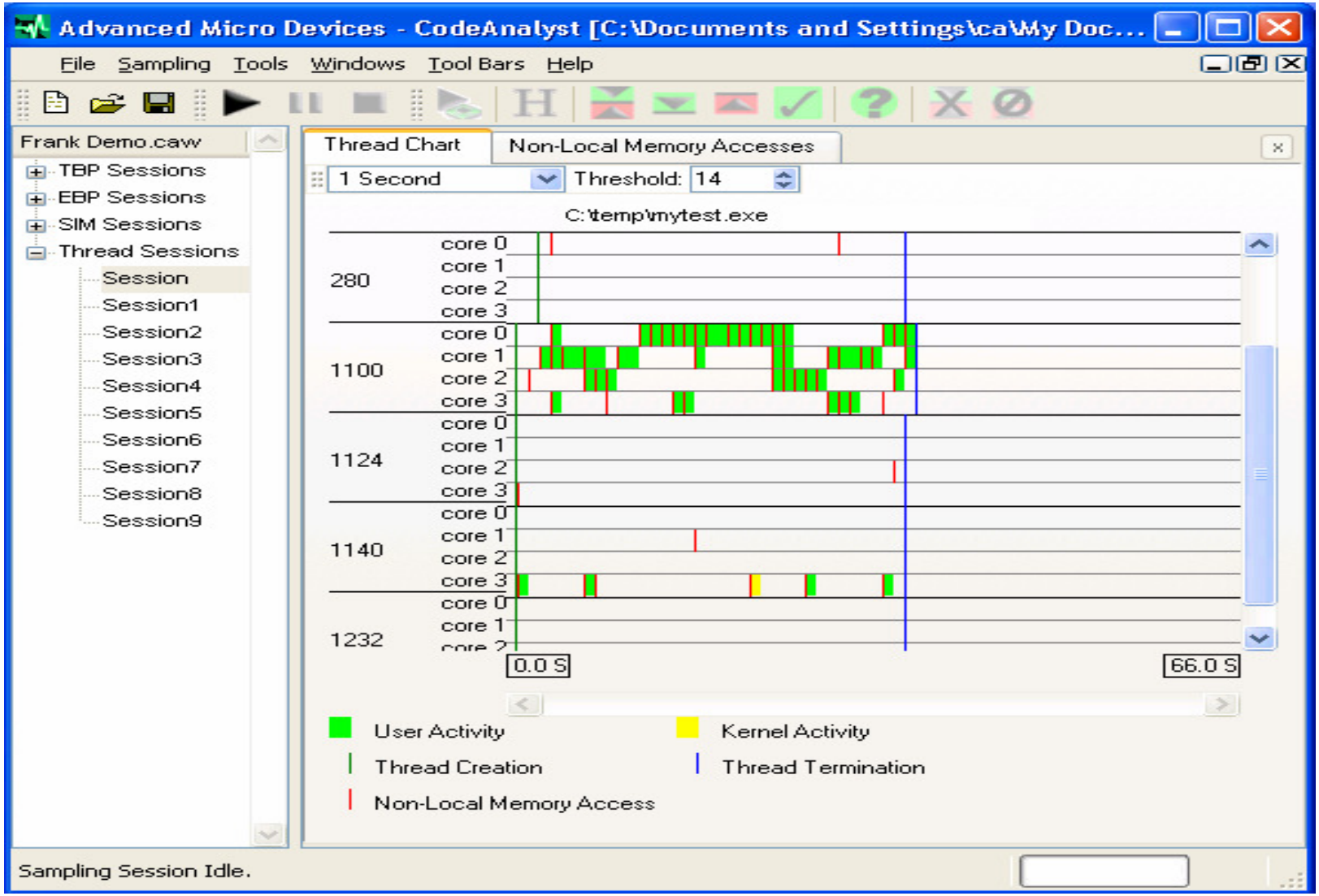


Thread Analysis

AMD
Smarter Choice

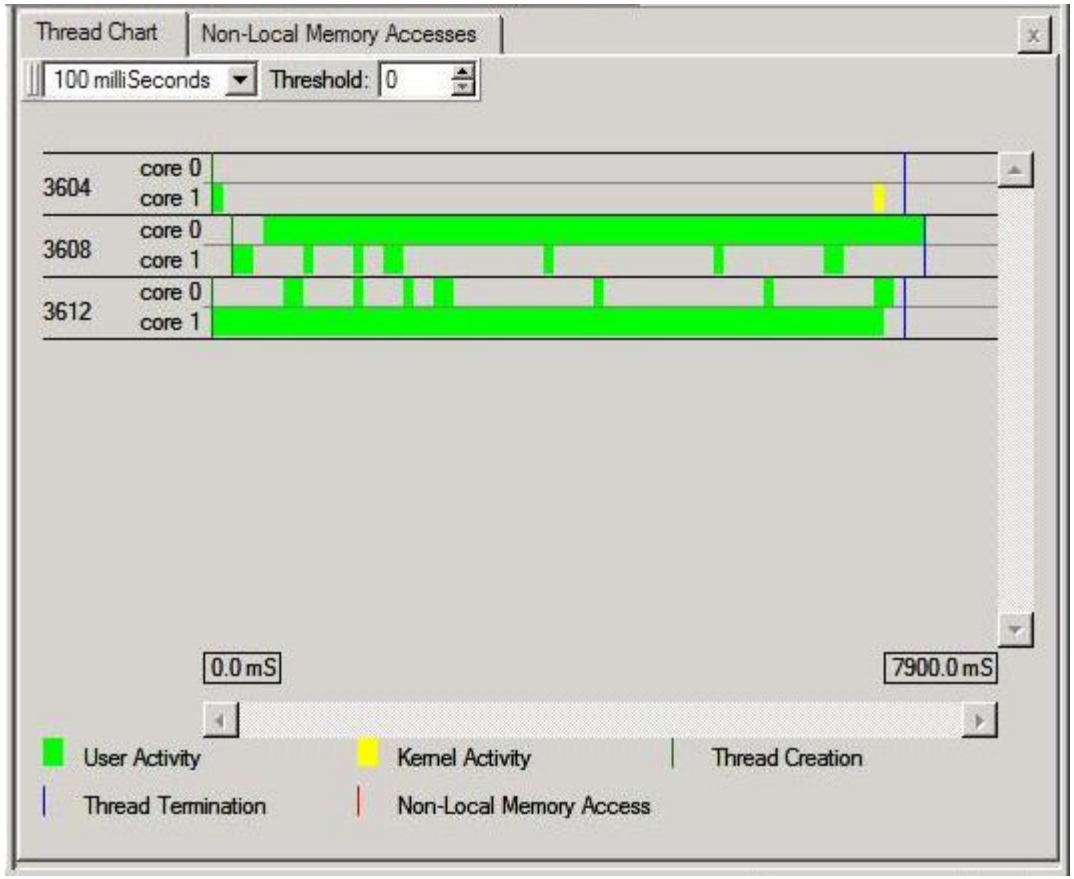
- Identities threads in the target application.
- Shows Thread creation and termination
- Monitors CPU affinity of each thread
- Identifies Non-local memory access
- Graphs thread activity on each CPU

Thread Analysis Data View



Thread analysis

- Display thread activity on cores
- Display remote/local memory access activity



Additional Resources

- CodeAnalyst is available to download at http://developer.amd.com/downloads.aspx
 We welcome feedback, bug reports, requests for help or new features.
 - You may email support.codeanalyst@amd.com

- Web Resources at:
 - http://www.developwithamd.com
 Software Optimisation Guide for AMD Athlon 64 and AMD Opteron
 AMD64 Architecture Programmer's Manual Volume 1: Application Programming
 AMD64 Architecture Programmer's Manual Volume 2: System Programming
 AMD64 Architecture Programmer's Manual Volume 3: General-Purpose and System Instructions
 AMD64 Architecture Programmer's Manual Volume 4: 128-Bit Media Instructions
 AMD64 Architecture Programmer's Manual Volume 5: 64-Bit Media and x87 Floating-Point Instructions
 - http://www.devx.com
 Optimizing Your C/C++ Applications, Part 1 & 2



Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice



AMD
Smarter Choice

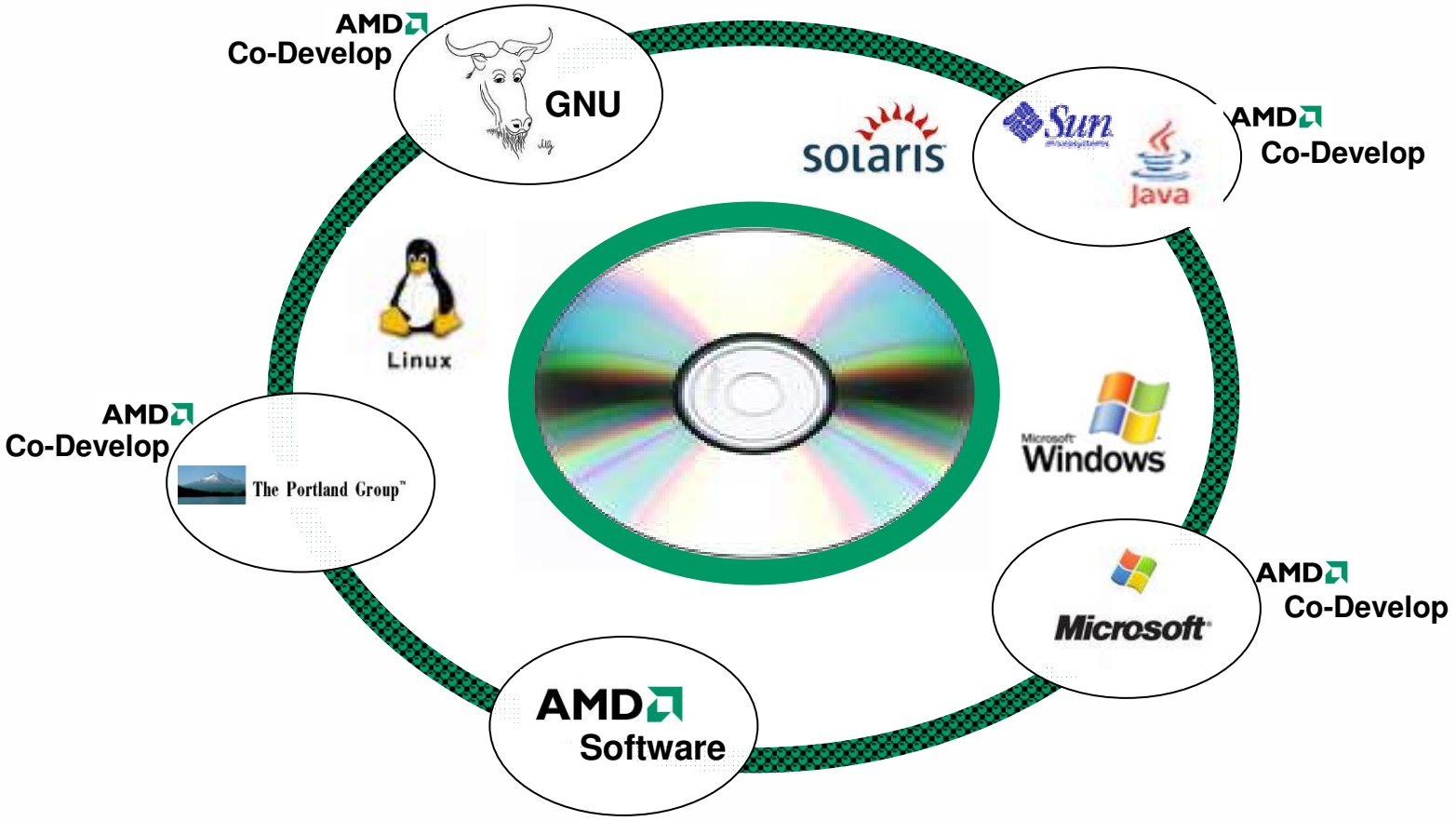
Compilers and Libraries

Hervé Chevanne – Dr. Ing.
Applications Analyst

Agenda

- AMD's Compiler Strategy
 - AMD64 & Java Update
- Leverage the Power
 - Single Machine
 - Distributed Computing
- Compiler Mechanisms and Libraries
 - Compiler Flags
 - ACML – AMD Core Math Library
- AMD Developer Resources
- Questions

AMD's Compiler Strategy
Provide Solutions to Customer Relevant Problems



Interoperability
Enabling software that works together

Overview of Compiler Capabilities

AMD
Smarter Choice

	Vector SIMD Support	Peels Vector Loops	Global IPA	Open MP	Links ACML Libraries	Profile Guided Feedback	Aligns Vector Loops	Parallel Debug-gers	Auto Paral-lelization	Medium Memory Model
PGI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GNU	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓
Microsoft	✗	✗	✓	✓	✓	✓	✗	✓	✗	✗
Sun	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Absoft	✓	✓	✗	✗	✓	✓	✓	✗		
Pathscale	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Intel	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓

Java & Enterprise Applications

- Work with IBM, Sun, and BEA to improve the performance of the Java runtimes on AMD platforms.
 - Feedback on codegen issues especially those related to AMD Architecture and Micro-architecture.
 - Feedback on platform issues such as Multi-core and NUMA
 - Use the fact that the language makes multithreading easy to study multithreaded performance issues
- Engage with Application Server vendors (e.g., WebSphere) to improve J2EE performance on AMD platforms.
 - Performance engineering to identify performance and scalability issues on AMD systems
 - Capture workloads of interest to processor/architecture development
 - Run performance analysis tests against a representative cross section of enterprise application architectures:
 N-tier Web, SOA, Others
 - New group just starting will grow to include at least one other application server (JBOSS or WebLogic).
- Developing XML processing benchmarks to evaluate and optimize software and hardware (Torrenza) solutions.

AMD64 Java Technology

J2SE 5.0: (Solaris, Linux, Windows)

- SUN JDK 5.0, JRE 5.0
- First version to support AMD64
- Latest JVM: 5.0 update 7

IBM Developer Kit for Java 5.0 (Linux, Windows)

- Latest JVM release: 5.0 update 3

BEA Jrockit JDK (Linux, Windows)

- Latest JVM release 5.0 R26 4.0

J2EE 1.4 SDK

Tools and Environment:

- Netbeans IDE (Windows, Solaris)
- Eclipse IDE (Windows, Linux)
- SUN Java Studio Creator 2
- SUN Java Studio Enterprise 8

AMD64 SUN JVM 1.5.x Optimisation

AMD Smarter Choice

Large Pages

Cross-platform flag: -XX:+UseLargePages (on by default for Solaris, off by default for Windows and Linux)

Linux and Windows (Server 2003) have to be configured to enable large pages

Solaris 9 onwards: Multiple Page Size Support (MPSS), allows a program to use any hardware supported page sizes to access portions of virtual memory.

Large Heap Size

32bit applications get limited to setting the heap below 4GB space.

64 bit doesn't have this limitation.

NUMA support

Various APIs (on Solaris and Linux: e.g. libnuma on Linux) that can be used by the processes themselves to select the best policy for a particular application.

In deployments that leverage multi-JVM, and where most of the data is not shared, assigning each JVM to a node can prove beneficial.

AMD64 SUN JVM 1.5.x Optimisation

AMD Smarter Choice

Generational Garbage Collection

Young Generation:

- The throughput collector uses a parallel collector (Flag: -XX:+UseParallelGC)
- Usually used with large number of processors.
- Throughput collector uses multiple threads for minor collections.
- For multi-JVM runs, these multiple GC threads can be restricted by using the flag:-XX:ParallelGCThreads=<desired number>).
- For applications that require larger young generation heap, use the flag Xmn to set the nursery.

Tenured Generation:

- The concurrent low pause collector is used to collect the tenured generation.
- The collection is done concurrently with the execution of the application.
- Short pauses are introduced during the collection (Flags: -XX:+UseConcMarkSweepGC)
- Use this collector when your application needs shorter garbage collector pauses and doesn't mind sharing processor resources with the GC.

AMD64 BEA R26.4.0 Optimisation

AMD
Smarter Choice

Large Pages

Cross-platform flag: -XXlargepages (off by default, enabled using XXaggressive as well)

Large Heap Size

32bit applications get limited to setting the heap below 4GB space.

64bit doesn't have this limitation, unless running with -XXcompresseRefs

NUMA support

Linux support for interleaving heap across nodes is integrated into JRockit.

Support includes node interleaving and binding of threads to CPUs

AMD64 IBM JVM 5 update 3 Opt.

AMD
Smarter Choice

Large Pages

Large pages support

Large Heap Size

32bit applications get limited to setting the heap below 4GB space.

64bit doesn't have this limitation

Garbage collection

Parallel generational GC

Concurrent generational GC

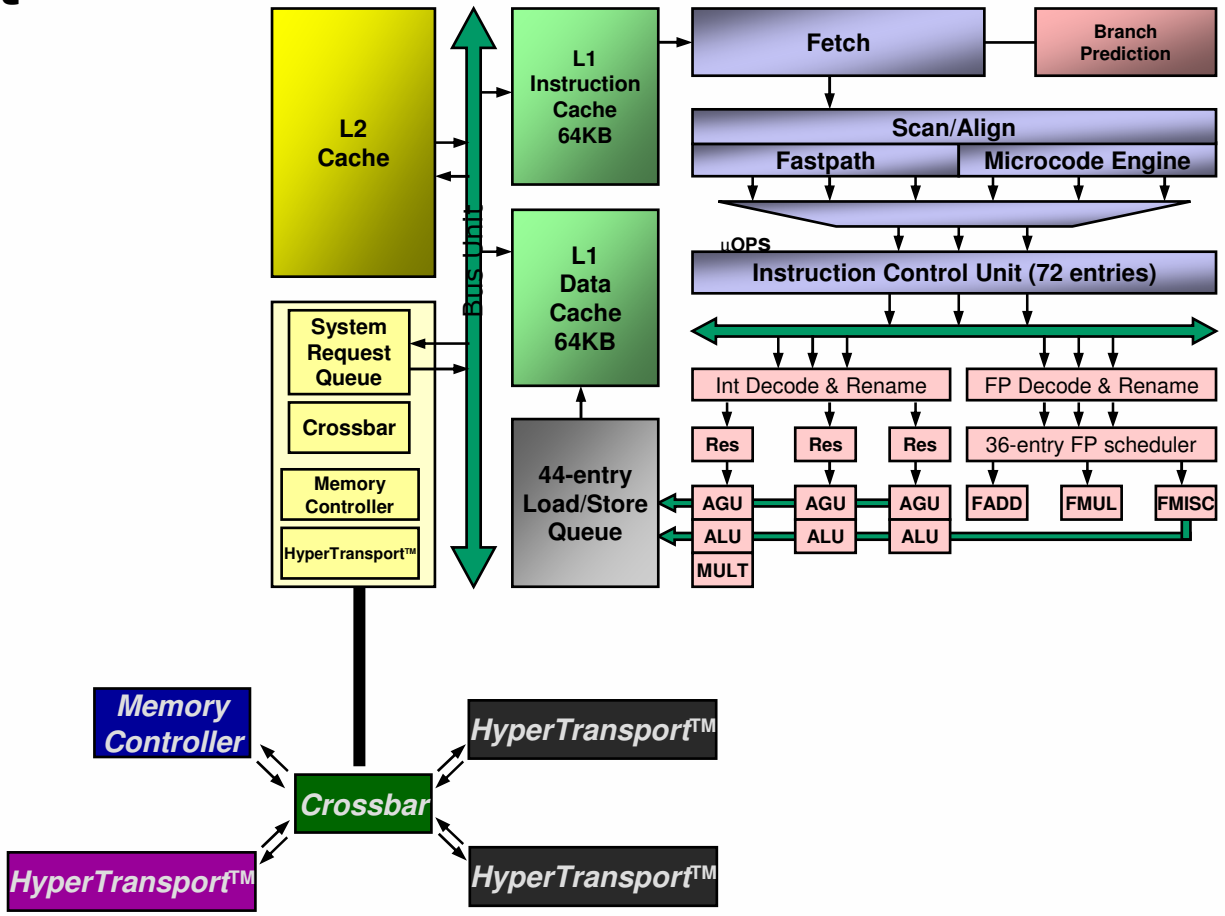


AMD
Smarter Choice

Leverage the Power

Leverage the Power

- **FPU Throughput**
- **32-bit Integer Throughput**
- **64-bit Integer Throughput**
- **6.4 GB/s per CPU of Memory Bandwidth**
- **25.6GB/s aggregate 4-socket Memory Bandwidth**

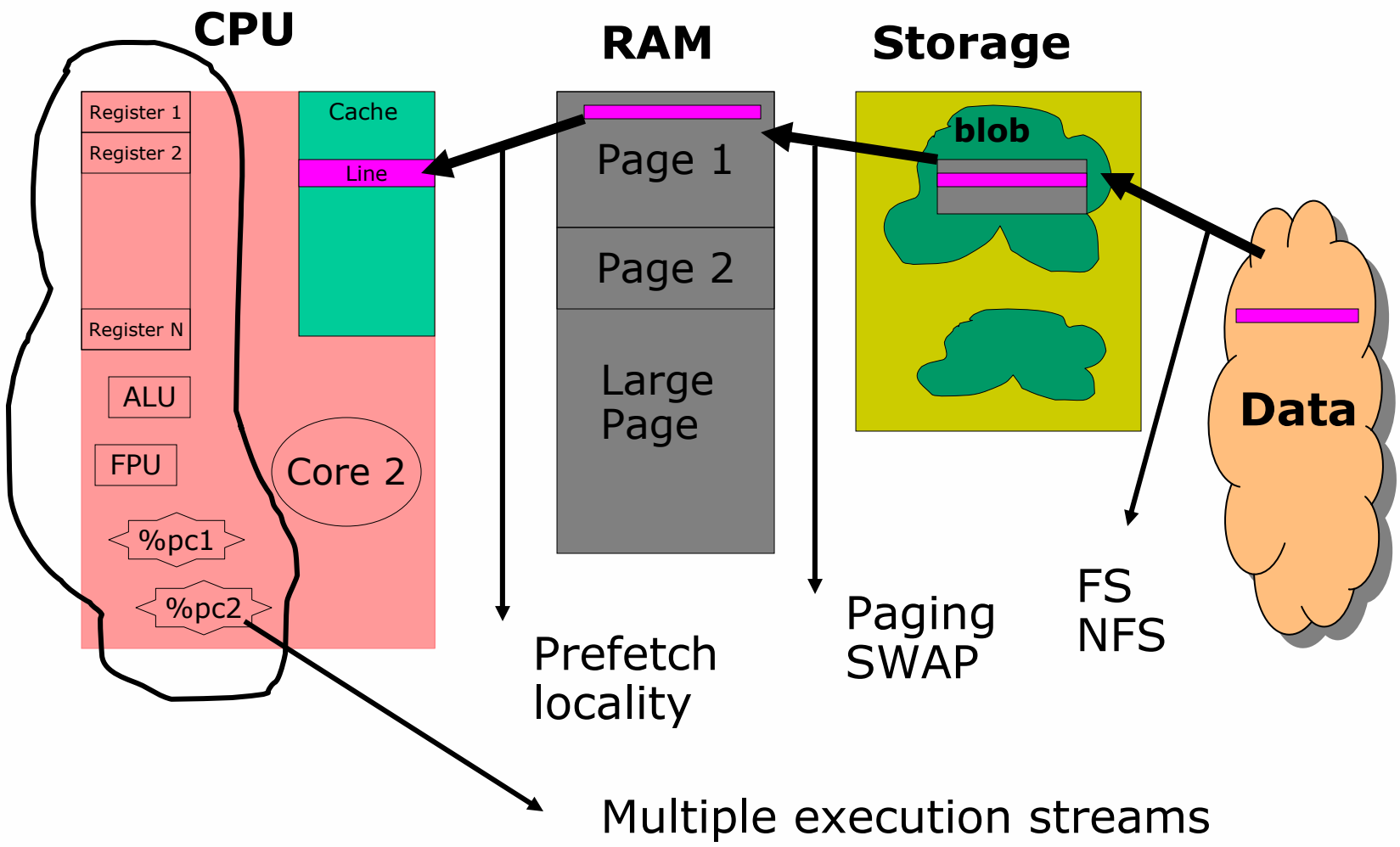


How do we take advantage of these capabilities ?

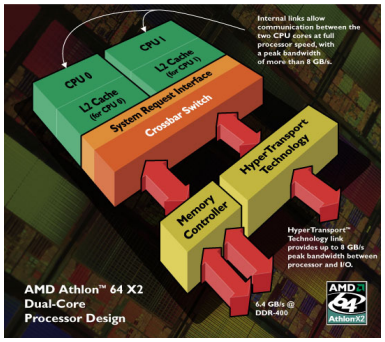
Leverage the Power

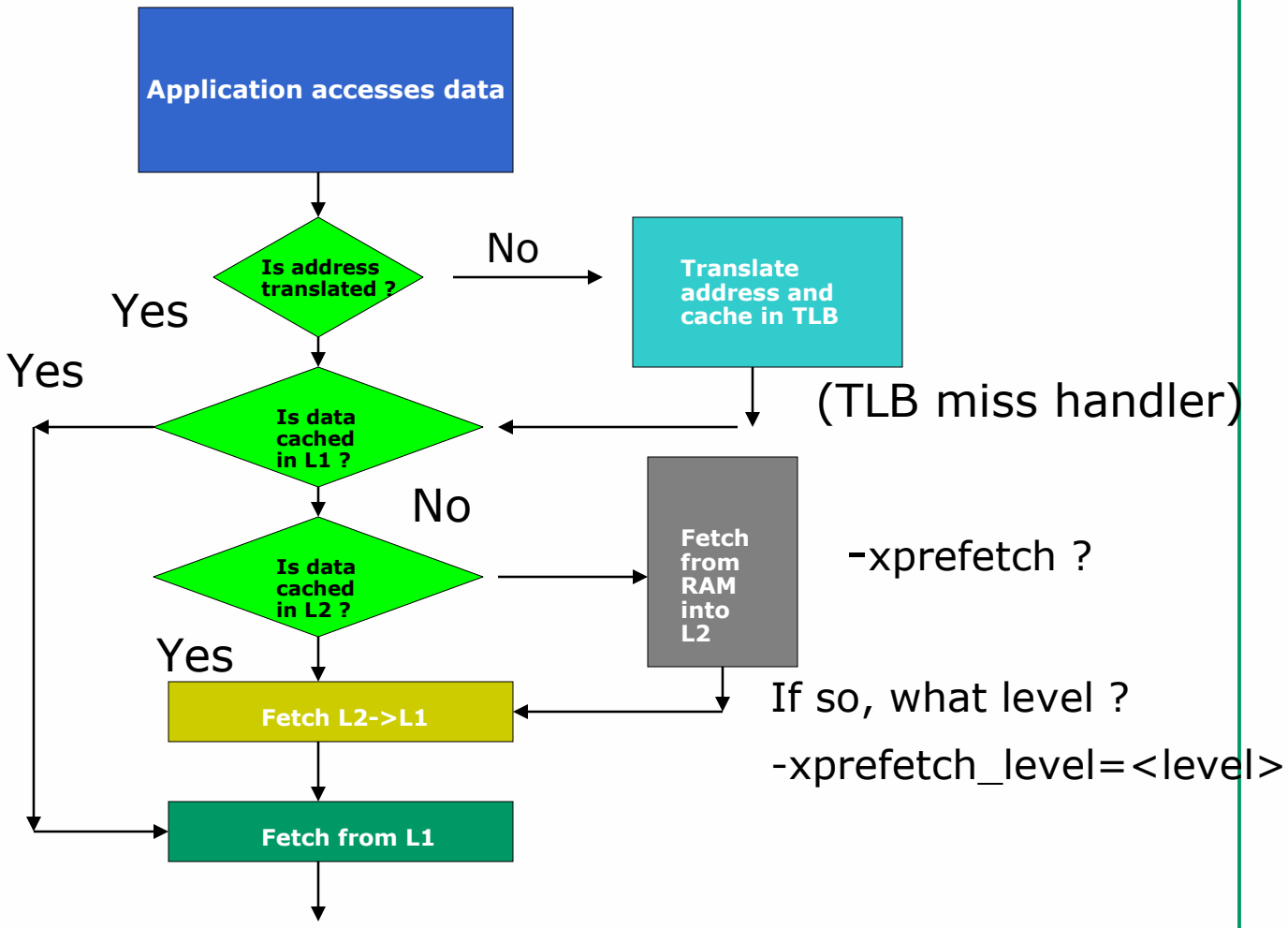
Switches, Options, Locality

AMD
Smarter Choice

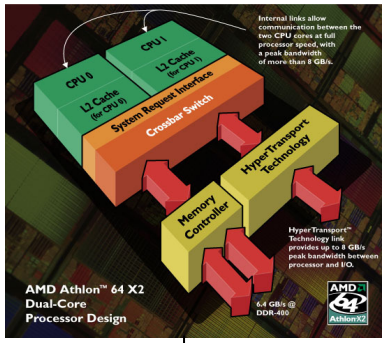


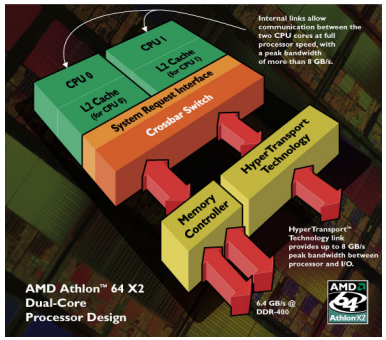
Leverage the Power
Under the Hood

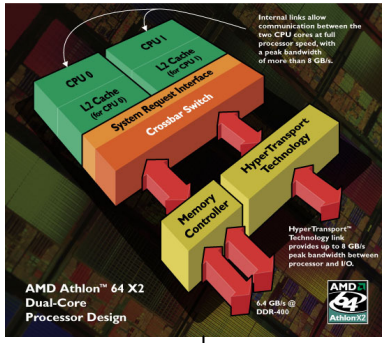




Leverage the Power
Distributed Computing

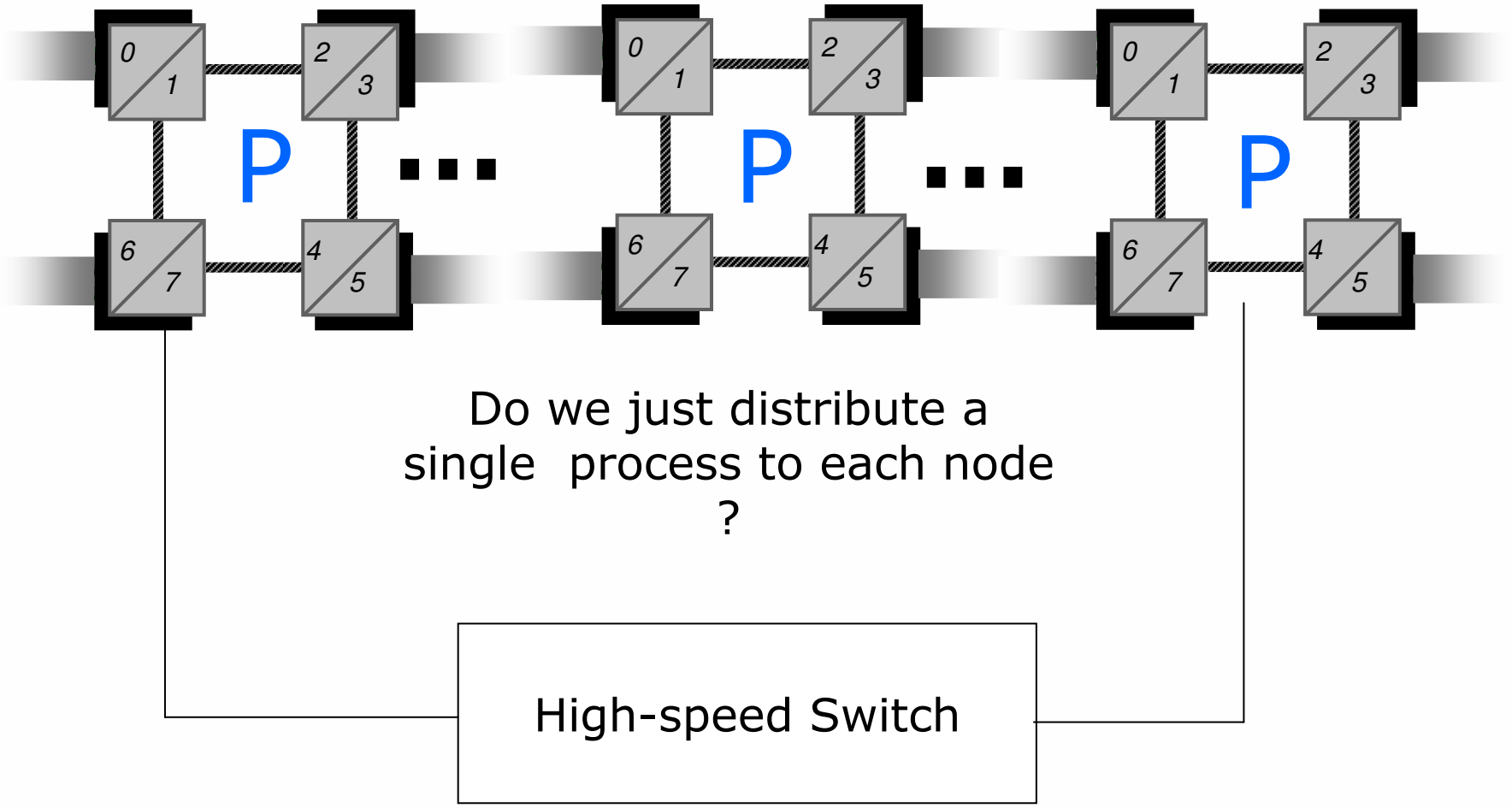






High-speed Switch

Leverage the Power
Distributed Computing





AMD
Smarter Choice

Compilers, Compiler Switches and Directives, and Libraries

Compilers
Platform availability matrix

AMD
Smarter Choice

Compiler Vendor	Compiler Platform		
	Linux	Microsoft® Windows®	Sun Solaris
PGI	**V6.2.5** *64-bit & 32-bit*	**V6.2.1** *64-bit & 32-bit*	
GCC	**V4.1.1 (FSF)** *64-bit & 32-bit*		**V4.1.1 (FSF)** *64-bit & 32-bit*
Microsoft®		**Visual Studio2005** *64-bit & 32-bit*	
Sun	**Sun Studio 11** *64-bit & 32-bit*		**Sun Studio 11** *64-bit & 32-bit*
PathScale	**V2.5** *64-bit & 32-bit*		
Intel	**V9.1** *64-bit & 32-bit*	**V9.1** *64-bit & 32-bit*	

*** Version numbers as of November 2006**

Approach: Ease of Concurrency

AMD Smarter Choice

- System Level:
 - **No change to application binary**. The OS will load applications to different processors so the throughput of the system as a whole will increase

 -mp=numa Links application with NUMA libraries

 MP_BLIST=3,2,1,0 Maps threads 0,1,2,3 to CPUS 3,2,1,0
- Application level:
 - **No Change to Code**

 Vectorization

 -Mvect=sse2

 Automatic parallelization

 -Mconcur on PGI: This supports data parallelization by splitting loop iterations over Processors/cores. Limited to loops or loop nests
 - **Change application**

 Thread *application using native calls (pthreads or Windows threads). The OS will schedule these threads to Processors/cores*

 Use OpenMp

 Standard for programming shared Memory architectures

 Directive driven and thus Portable

 Supports both data parallelism and functional parallelism paradigms

 Use of Multithreaded libraries, e.g., ACML

 Code that uses libraries can use multithreaded version of the libraries for performance gain.

Gains From Tuning

AMD
Smarter Choice

Tuning Category	Typical Range of Gain
Source Change	25-300%
Compiler Flags	5-20%
Use of libraries	25-200%
Assembly coding/tweaking	5-20%
Manual prefetching	5-30%
TLB thrashing/cache	20-100%
Using SSE/inlines/micro-vectorization	25-200%

Let's Look at Low Hanging Fruit first

Common Loop Transforms

AMD
Smarter Choice

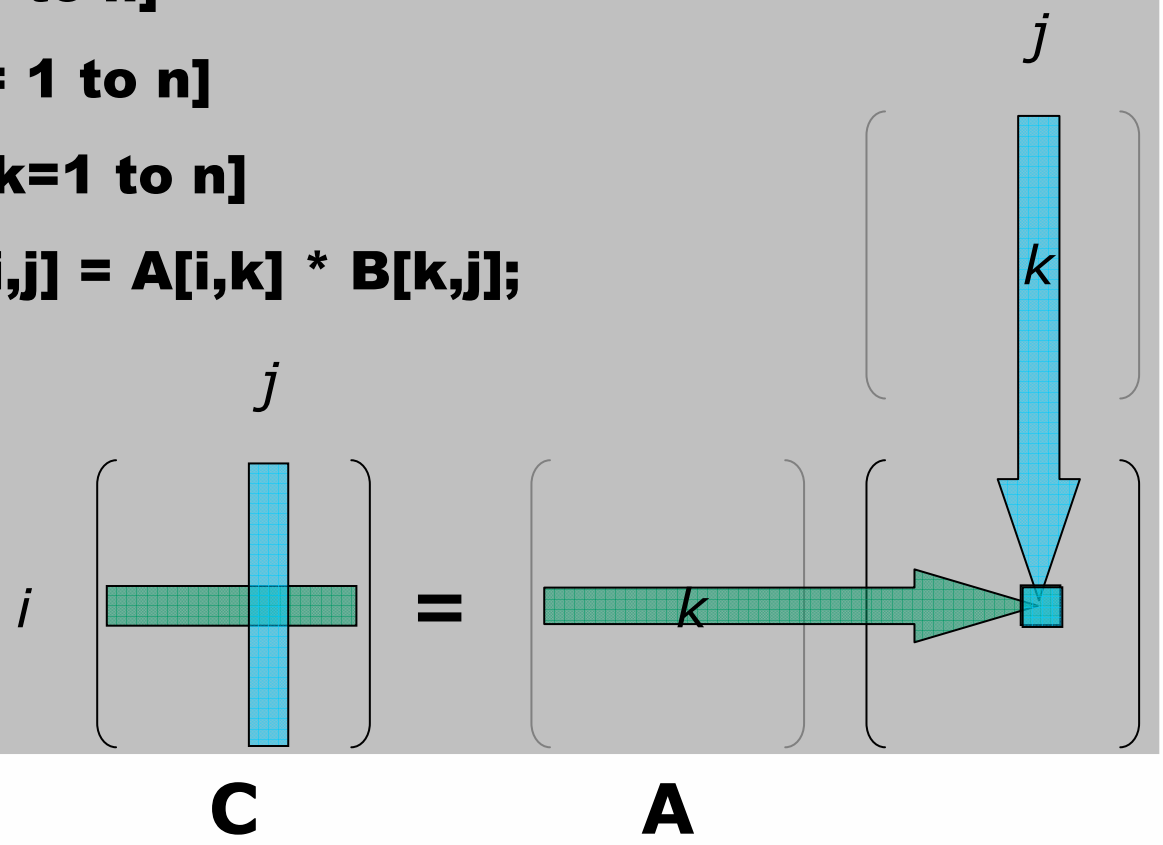
- **Loop Interchange**: Interchange outer and inner loops
- **Loop Distribution**: Split one loop into two
- **Loop Fusion**: Combine two loops into one
- **Loop Unrolling**: Replicate loop body and do fewer iterations
- **Strip Mining**: Divide iterations of one loop into two nested loops
- **Unroll and Jam**: Combine unrolling, strip mining and interchange
- **Loop blocking**: Divide iteration space into rectangular blocks
- **Scalar Replacement**: Eliminate load/store by reusing values across innermost loop iterations
- **Loop Peeling**: Peel off iteration to align data

Example: Unroll & Jam Matrix Multiply

AMD Smarter Choice

$$[C] = [A] * [B]$$

```
for [i = 1 to n]
   for [j = 1 to n]
      for [k=1 to n]
         C[i,j] = A[i,k] * B[k,j];
```



Example: Unroll & Jam Matrix Multiply

AMD Smarter Choice

```
for [i = 1 to n by 2]
   for [I = i to i+1]
      for [j = 1 to n]
         for [k = 1 to n]
            C[I,j] = A[I,k] * B[k,j];
```

Unroll once and

divide i^{th} iterate into two

Example: Unroll & Jam Matrix Multiply

AMD Smarter Choice

Interchange strip mined loop to inside

```
for [i = 1 to n by 2]
   for [j = 1 to n]
      for [k = 1 to n]
         for [I = i to i+1]
            C[I,j] = A[I,k] * B[k,j];
```

One half as many iterations

Strip of two iterations

Example: Unroll & Jam Matrix Multiply

AMD Smarter Choice

```
for [I = 1 to n by 2]
    for [j = I to n]
        for [k = 1 to n]
            C[I,j] = A[I,k] * B[k,j];
            C[I+1, j] = A[I+1,k] * B[k,j];
```

One half as many iterations

Twice as much parallelism

Reuse B[k,j]

Example: Peels loops

AMD
Smarter Choice

Assume aligned data

```
Double a[100], b[100], c[100];
  for [i=1 to 100]
    a[i] = b[i] - c[i];
```

Aligned on 16 byte boundaries

Example: Peels loops

AMD Smarter Choice

Compiler can use "move two aligned double precision values to XMM register" instruction

```
Double a[100], b[100], c[100];
  for [i=1 to 100]
    a[i] = b[i] - c[i];
```

Aligned on 16 byte boundaries

Is "vectorized" into

```
SUB:
  movapd    xmm0,b[ecx]
  subpd     xmm0, c[ecx]
  movapd    a[ecx], xmm0
  Add       ecx, 16,
  Cmp       ecx, 800
  J1        SUB
```

Load two dp fp #s

Subtract two dp fp #s

Store two dp fp #s

Stride is 16 since can access two dp fps at once

Test at end of array, 8bytes * 100 = right ?

Example: Peels loops

AMD Smarter Choice

If unaligned, compiler will "peel" off one loop, and then align rest of array on a 16 byte boundary

```
Double a[100], b[100], c[100];
  a[1] = b[1] - c[1]
  for [i=1 to 100]
    a[i] = b[i] - c[i];
```

Unaligned on 16 byte boundaries

In this case a static alignment Optimisation was used. For cases that cannot be determined at runtime, a dynamic alignment Optimisation is used that tests a few "peels" before an access pattern is detected

Tuning Performance with Compilers
Maintaining Stability while Optimizing

AMD
Smarter Choice

- STEP 0: Build application using the following procedure:

compile all files with the most aggressive Optimisation flags below:

```
-tp k8-64 -fastsse -Mipa=fast,inline
```

- STEP 1: Profile binary and determine performance critical routines

- STEP 2: Repeat STEP 0 on performance critical functions, one at a time, and run binary after each step to check stability

PGI Compiler Flags
Optimisation Flags

AMD Smarter Choice

Below are 3 different sets of recommended PGI compiler flags for flag mining application source bases:

- **Most aggressive: -tp k8-64 –fastsse –Mipa=fast,inline**

 enables instruction level tuning for Opteron, O2 level Optimisations, sse scalar and vector code generation, inter-procedural analysis, LRE Optimisations and unrolling

 strongly recommended for any single precision source code

- **Middle of the ground: -tp k8-64 –fast –Mscalarsse**

 enables all of the most aggressive except vector code generation, which can reorder loops and generate slightly different results

 in double precision source bases a good substitute since Opteron has the same throughput on both scalar and vector code

- **Least aggressive: -tp k8-64 –O0 (or –O1)**

-tp k8-64e: *SSE3 capable Opteron™ proc. (rev. E+)*

-tp k8-64: *older versions*



PGI Compiler Flags
Functionality Flags

AMD
Smarter Choice

- **-mcmodel=medium**

 use if your application statically allocates a net sum of data structures greater than 2GB

- **-Mlarge_arrays**

 use if any array in your application is greater than 2GB

- **-mp**

 process OpenMP/SGI directives/pragmas (build multi-threaded code)

- **-Mconcur(=dist:block)**

 attempt auto-parallelization of your code on multicore or SMP system with OpenMP

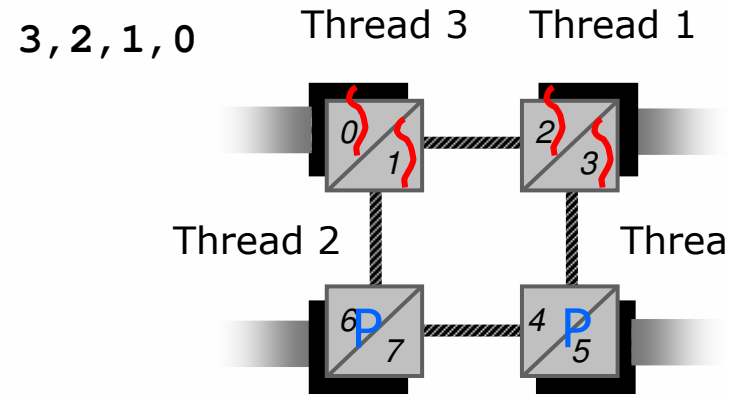


PGI Compiler for Dual Cores
Controlling CPU Core/Memory Affinity

PGI 6 allows users to control process/core affinity in OMP applications when compiled with –mp

PGI 6.2
-mp=numa

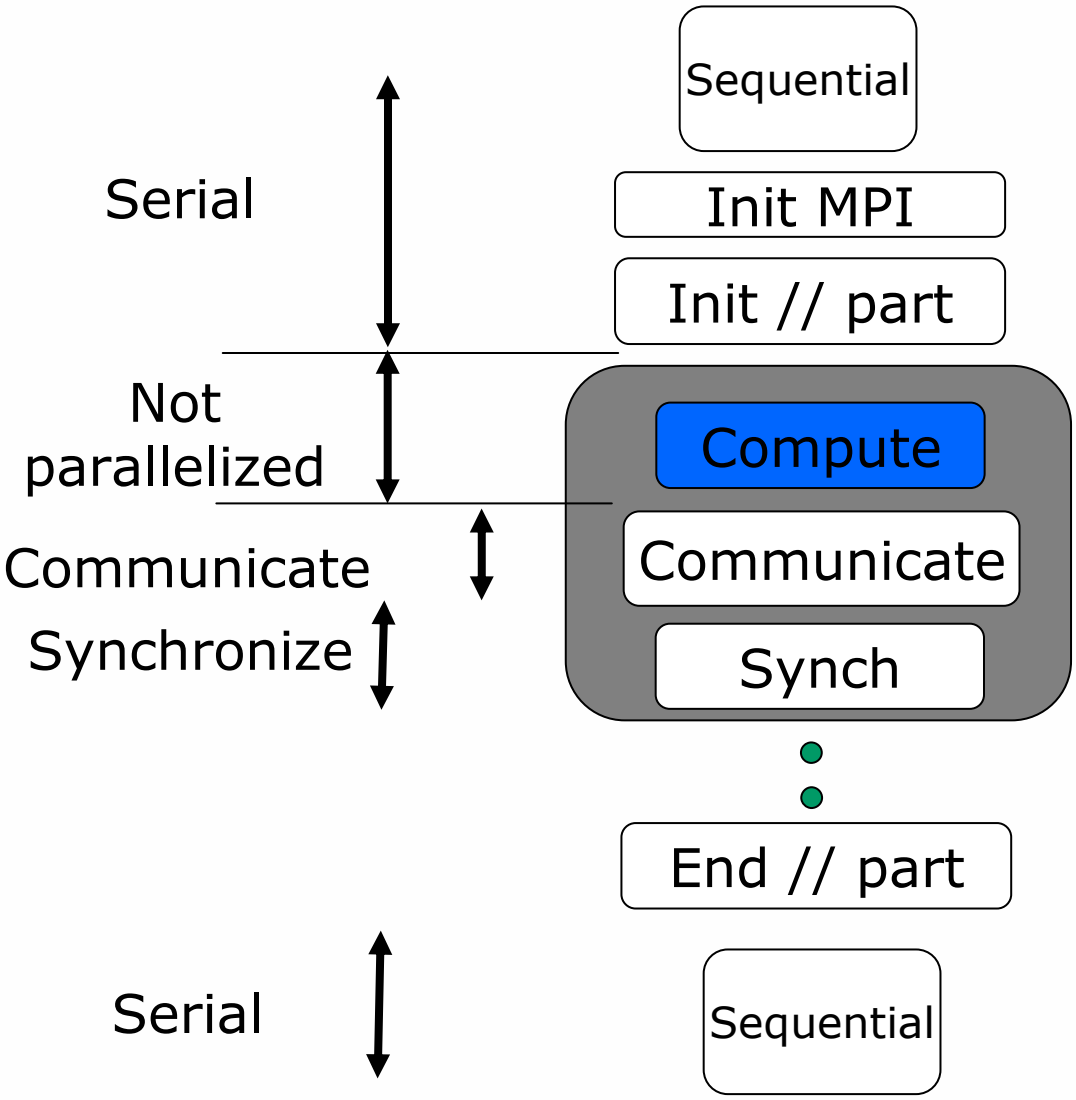
- **Environment variables required for OMP and NUMA**
 - `> setenv OMP_NUM_THREADS 4`
 - `> setenv MP_BIND yes`
 - `> setenv MP_BLIST 3,2,1,0`



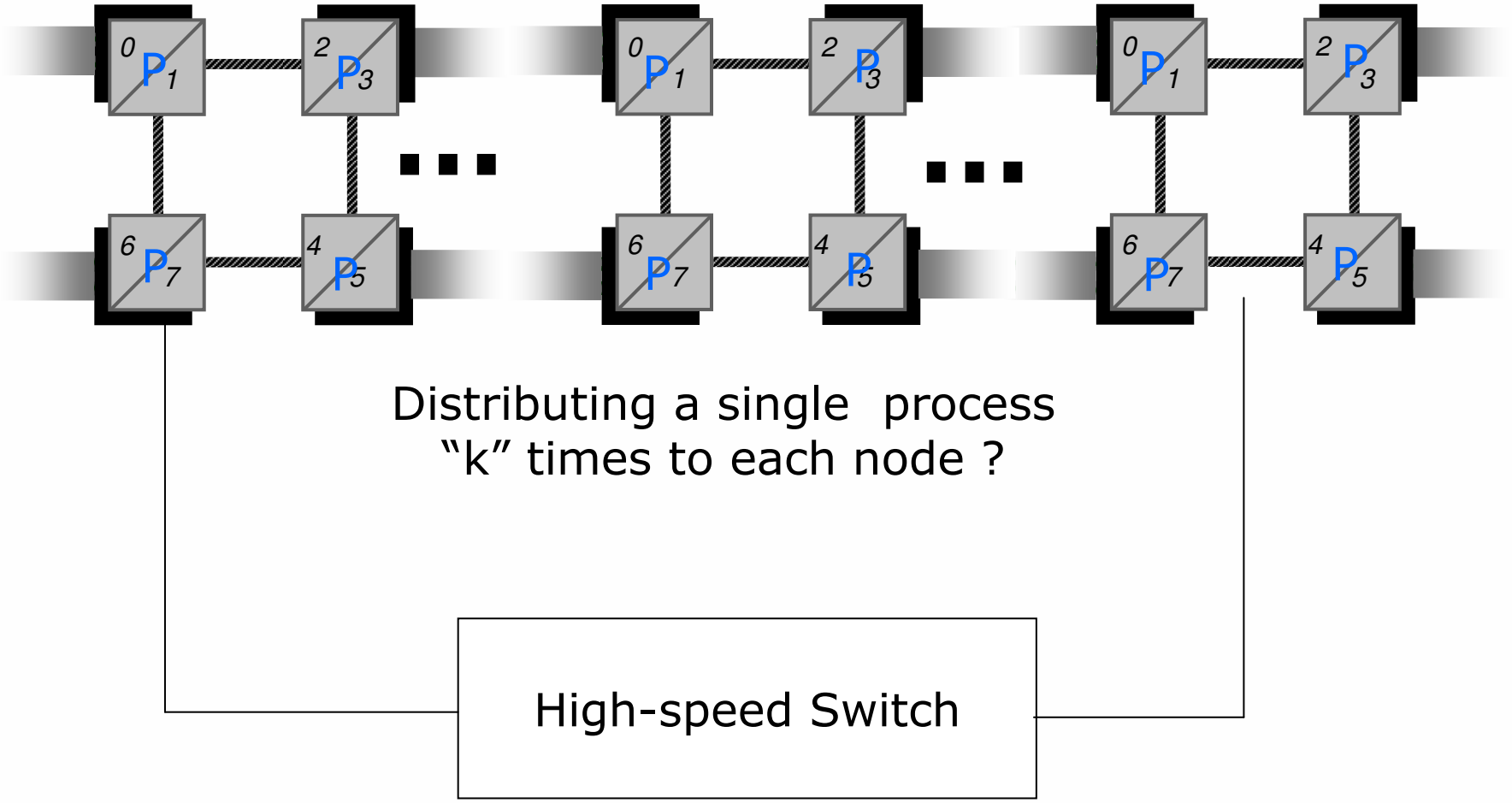
Interleaves threads 0,1,2,3 on cores 3,2,1,0



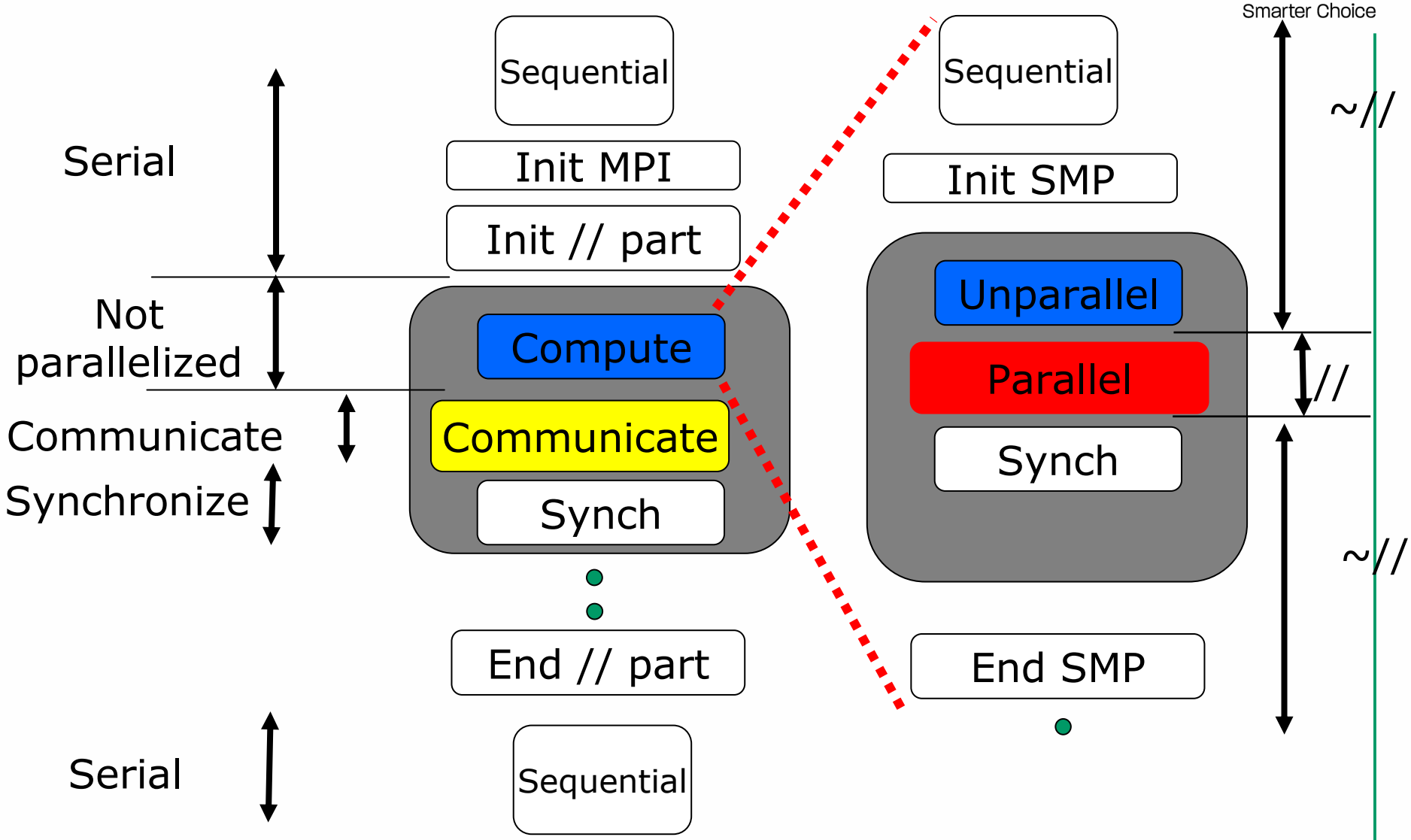
Message Passing Interface Model



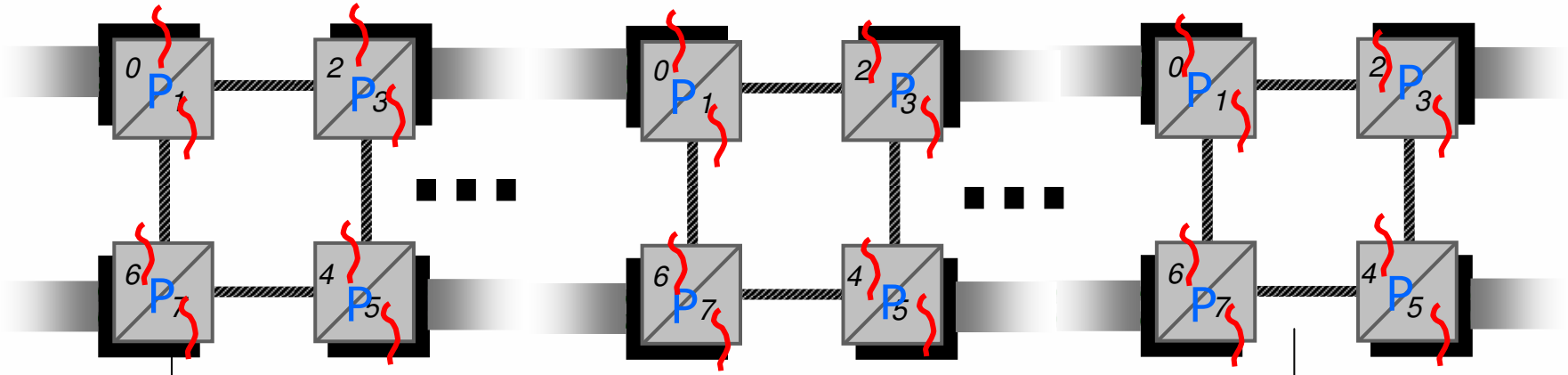
Leverage the Power Distributed Computing



MPI + OpenMP


Sequential
Init MPI
Init // part
Compute
Communicate
Synch
End // part
Sequential
Serial
Not parallelized
Communicate
Synchronize
Serial
Sequential
Init SMP
Unparallel
Parallel
Synch
End SMP
~//
//
~//

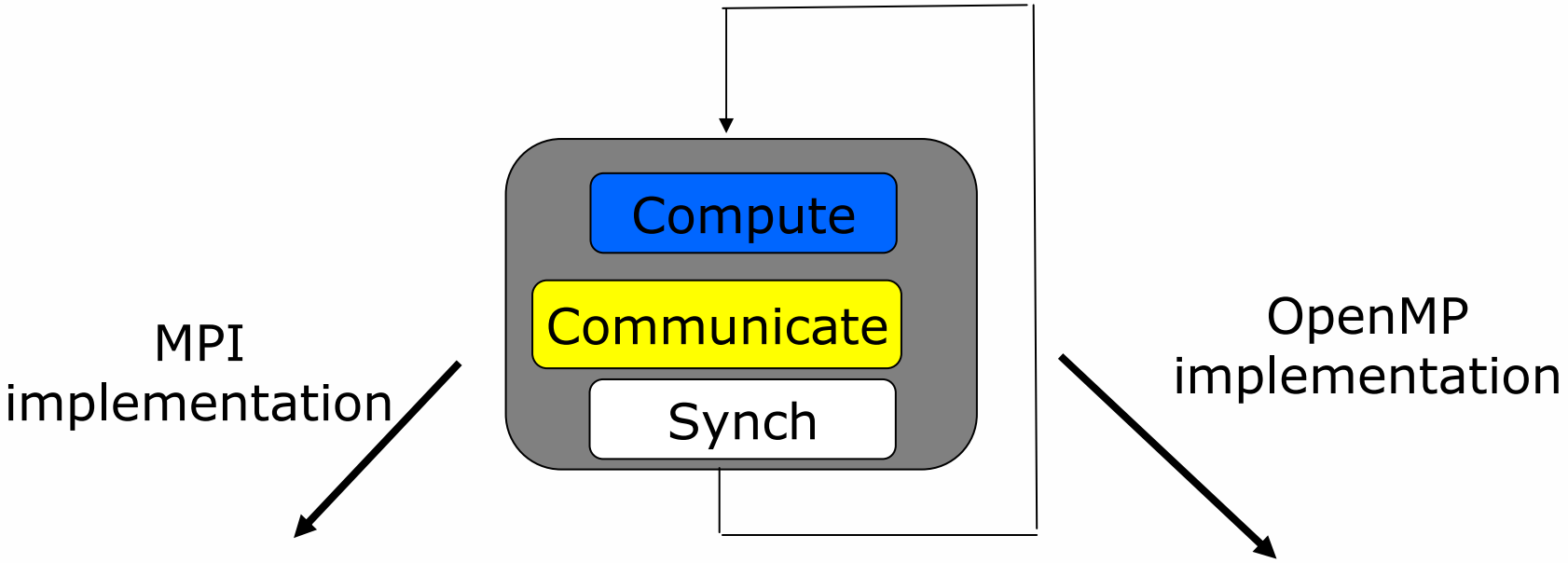
Leverage the Power Distributed Computing



Distributing k processes to each node that in turn spawn "n" OpenMP threads

High-speed Switch

Communication in MPI and OpenMP




```
FOR p=1 TO <all other processors>
    Send(p, Work(…p…))
```

```
FOR p=1 TO <all other processors>
   COPY Work(…p…)->GlobalBuffer(…p…)
BARRIER
FOR p=1 TO <all other processors>
  COPY GlobalBuffer(…p…)-> Work(…p…)
```

Example: Parallel Conjugate Gradient

AMD Smarter Choice

```
For [i=1 to n Processors]
   dot=(res(i),res(i))
   v1(i) = daxpy(A,x(i))
   grad(i) = v1(i)-b
   alpha = dot1/(res(i+1),res(i+1))
   dir(i+1) = -grad(i) + alpha*dir(i)
   v2(i+1) = daxpy(A,dir(i+1))
   step = (dir(i+1),dir(i+1))/(dir(i+1),v2(i+1))
   xiterate(i+1) = xiterate(i) +step*dir(i+1)
Enddo
```

Parallel Vector dot product

Parallel Vector dot product

Example: Parallel Conjugate Gradient

AMD Smarter Choice

```
For [i=1 to n Processors]
   omp_set_num_threads (atoi[argv[3]]);
   dot=(res(i),res(i))
   v1(i) = daxpy(A,x(i))
   grad(i) = v1(i)-b
   alpha = dot1/(res(i+1),res(i+1))
   dir(i+1) = -grad(i) + alpha*dir(i)
   v2(i+1) = daxpy(A,dir(i+1))
   step = (dir(i+1),dir(i+1))/(dir(i+1),v2(i+1))
   xiterate(i+1) = xiterate(i) +step*dir(i+1)
Enddo
```

Specify # threads/process

Parallel Vector dot product

Parallel Vector dot product

And in daxpy, use OpenMP directive to parallelize the loop

AMD Smarter Choice

```
daxpy(matrix a, vector b)
#pragma omp parallel for private(j,tmp)
 FOR [i = 1; i < BLOCK_SIZE(id,p,n); i++]
    tmp = 0.0
    for [j = 1; j < n+1; j++]
       tmp += a[i][j] * b[j]
       MPI_part_of_vector[i] = tmp
   END J
END I
```

Add this directive

ACML- AMD Core Math Library

AMD
Smarter Choice

- **Suite of highly tuned math functions for high performance computing**

- **BLAS** – Basic Linear Algebra Subprograms
 - Full Level 1, 2, and 3 support
 - Highly optimized DGEMM, other Level 3 BLAS.
 - OpenMP support for key routines

- **Lapack** – Linear Algebra package
 - Uses calls to BLAS to solve linear algebra systems
 - Given Matrix A, Vectors x,y, Solve A*x =y for x
 - OpenMP support for key routines

- **FFTs** – Fast Fourier Transforms
 - Time-to-frequency domain
 - Hand-tuned assembly
 - OpenMP support for 2D, 3D transforms

- **Fast Math and Fast Vector Math Library**

- **RNGs** -Random Number Generators
 - Comprehensive reference implementation
 - Tuning required

Microsoft Windows

AMD Smarter Choice

# of processors	OS Bits	Compilers	ACML install directory
Single	64	PGI pgf77/pgf90/pgcc/MSC	acml3.5.0/win64
Multi	64	PGI pgf77/pgf90/pgcc/MSC	acml3.5.0/pgi64_mp
Single	32	PGI pgf77/pgf90/pgcc/MSC	acml3.5.0/pgi32
Multi	32	PGI pgf77/pgf90/pgcc/MSC	acml3.5.0/pgi32_mp

PGI Compiled Version

- For 32-bit Windows, SSE and SSE2 instruction support is required
- For 32-bit Windows, DLL only is provided
- For 64-bit Windows, DLL and LIB are provided

Where to Get

- ACML
 - publicly released via the Web
 www.amd.com/acml
 - User registration and sign in required for download
 - Support
 acml.support@amd.com
 devforums.amd.com

Microsoft Compiler Optimisation Flags

- **/O2 /Ob2 /GL /fp:fast improve performance**

/O2 turns on several general Optimisations.

/Ob2 turns on inline expansion

/GL enables inter-procedural Optimisations

/fp:fast allows the compiler to use a fast floating-point model

- **/GS-**

***/GS** attempts to detect buffer overruns in the return address (default)*

- **/D_ST_MODEL**

Turn off default compiler behavior of emitting thread-safe code

- **Feedback Directed Optimisation**

STEP 0**: Compile binary with **/GL** switch and link with **/LTCG

***STEP 1**: Run code collect data*

STEP 2**: Relink the program with **/LTCG:PGO

Microsoft Compiler Functionality Flags

- **/GR**

 Enables run-time information

- **/GT**

 Supports fiber safety for data allocated using static thread-local storage

- **/Wp64**

 Detects most 64-bit portability problems

- **/LD**

 Creates a dynamic-link library

- **/Oa**

 Assumes no aliasing

- **/Ow**

 Assumes aliasing across function calls but not inside functions

GCC Compiler Flags - Optimisation Flags

AMD Smarter Choice

- Below are 3 different sets of recommended gcc compiler flags for flag mining application source bases:
- Most aggressive: -O3 -ffast-math -funroll-all-loops -fpeel-loops -ftracer -funswitch-loops –ftree-vectorize –mtune=k8
 - General Optimisations, fast floating point model,-funroll-all-loops causes all loops to be unrolled (makes code larger!), and vectorizations, and tunes for Opteron and Athlon.
- Middle of the ground: -O2 –mtune=generic
 - Will optimize fairly well for Opteron/Athlon as well as Intel processors and should produce reliable code.
- Least aggressive: O1
- Blended Code Future
 - Allows good performance on all x86_64 vendor processors
 - Blended code model checked in January 19th 2006 for 4.2 of GCC (joint work with SUSE)
 - -mtune=generic32 for 32-bit mode
 - -mtune=generic64 for 64-bit mode
 - Use -mtune=generic for 4.1 of GCC and earlier



Linux64: GNU Compiler Flags Optimisation Flags (64 bit)

AMD
Smarter Choice

Compiler version	Languages	Recommended Optimisation Switches
SuSE gcc 3.3.3 **Red Hat gcc-sa**	C, C++, Fortran	-O3 -ffastmath -funroll-all-loops
SuSE gcc 3.3.4		-O3 -ffast-math -funroll-all-loops -fpeel-loops
FSF gcc 3.4 **Red Hat gcc 3.4.1**	C, C++, Fortran	-O3 -ffast-math -funroll-all-loops -fpeel-loops -ftracer -funswitch-loops -funit-at-a-time
FSF gcc 4.0+	C, C++, Fortran	-O3 -ffast-math -funroll-all-loops -fpeel-loops -ftracer -funswitch-loops -funit-at-a-time -ftree-vectorize



Solaris: Sun Microsystems Compiler Flags Optimisation Flags

AMD Smarter Choice

- **Most aggressive: -fast -xarch=amd64** (and not vice versa)

 f95 equivalent: -O5 -xarch=sse2 -xcache=64/64/2:1024/64/8 -xchip=opteron -dalign -fsimple=2 -fns=yes -ftrap=common -xlibmil -xlibmopt -nofstore -xarch=amd64

- **Less aggressive:**

 *In some cases **–O2** or **–O3** is faster than **–O5** or **–O4***

 -fsimple=0 Permit no simplifying assumptions. Preserve strict IEEE 754 conformance.

 -fsimple=1 Allow conservative simplifications. The resulting code does not strictly conform to IEEE 754, but numeric results of most programs are unchanged.

 -fsimple=2 Permit aggressive floating point Optimisations. This can cause some programs to produce different numeric results due to changes in the way expressions are evaluated.

```
> f95 -fast -xarch=amd64 -fsimple=1 -xnolibmopt ...
```

- **Least aggressive: -O1 -xarch=amd64**

Sun microsystems

Compiler Options for Performance

AMD Smarter Choice

- Compiler Optimisation options:
 - -xO1 thru -xO5 (default is "none", -O implies -xO3)
 - -fast:easy to use, best performance on most code, but it assumes compile platform = run platform and makes FP arithmetic simplications
 - Understand program behavior and assert to optimizer:

 -xrestrict, if only restricted pointers are passed to functions

 -xalias_level, if pointers behave in certain ways

 -fsimple if FP arithmetic can be simplified
 - Target machine-related:

 -xprefetch, -xprefetch_level

 -xtarget=, -xarch=, -xcache=, -xchip=

 -xvector to convert DO loops into vector instr/calls

Sun microsystems

Compiler Options for Performance

AMD Smarter Choice

- Advanced compiler options

 -xprofile: profile-feedback enabled Optimisations

 -xcrossfile, -xipo: performs crossfile/interprocedural Optimisations

 -xautopar: enable automatic parallelization

 -xdepend: performance dependence analysis

- Use optimized libraries

 Use Sun Performance library for algebraic functions

 Use libmvec for vectorized math routines

 -xlibmil, -xlibmopt for inline and optimized math

 Use value-added math library: libsunmath

Sun microsystems

Compiler Options for Performance

AMD Smarter Choice

- Advanced compiler options

 - **-xmodel=medium**
 use if static data structures are greater than 2GB
 - **-dy/dn**
 allow or disallow dynamic libraries for the entire executable (defaut is –dy)
 - **-C**
 enable runtime subscript range checking
 - **Feedback Directed Optimisation**
 STEP 0: Compile binary with -xprofile=collect
 STEP 1: Run code collect data
 STEP 2: Recompile binary with -xprofile=use
 - **-xtarget=(generic|generic64|native|native64)**
 default is generic

Sun microsystems

Sun Performance library

AMD Smarter Choice

- Numerical routines optimized for performance, including:
 - LAPACK v3.0
 - BLAS 1, 2, 3
 - Netlib Sparse BLAS
 - NST Fortran Sparse BLAS v0.5
 - Fast Fourier Transform (FFT) routines
 - Convolution and correlation routines
- Support for single- or multi-processor systems
- Hotspot tuning for BLAS routines
- Routines are optimized for various SPARC chips, X86 and AMD64 platforms

Sun microsystems

Source code Optimisations

AMD
Smarter Choice

- Improve usage of data cache, TLB
- Optimize data alignment (also: #pragma align, -dalign)
- Prevent Register Window overflow
- Creating inline assembly templates for performance critical routines
- Loop Optimisations that compilers may miss:
 - Restructuring for pipelining and prefetching
 - Loop splitting/fission
 - Loop Peeling
 - Loop interchange
 - Loop unrolling and tiling
 - Pragma directed





AMD
Smarter Choice

CodeAnalyst Demo

Hervé Chevanne
Justin Boggs

Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

AMD
Smarter Choice

REMINDER
Turn in the attendee survey for your gift.



AMD
Smarter Choice

Open Q&A

Hervé Chevanne
Justin Boggs

AMD
Smarter Choice

Thank you!

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2006 Advanced Micro Devices, Inc. All rights reserved.

Tools and resources

to make the

Smarter Choice



developer.amd.com

Because platform matters.

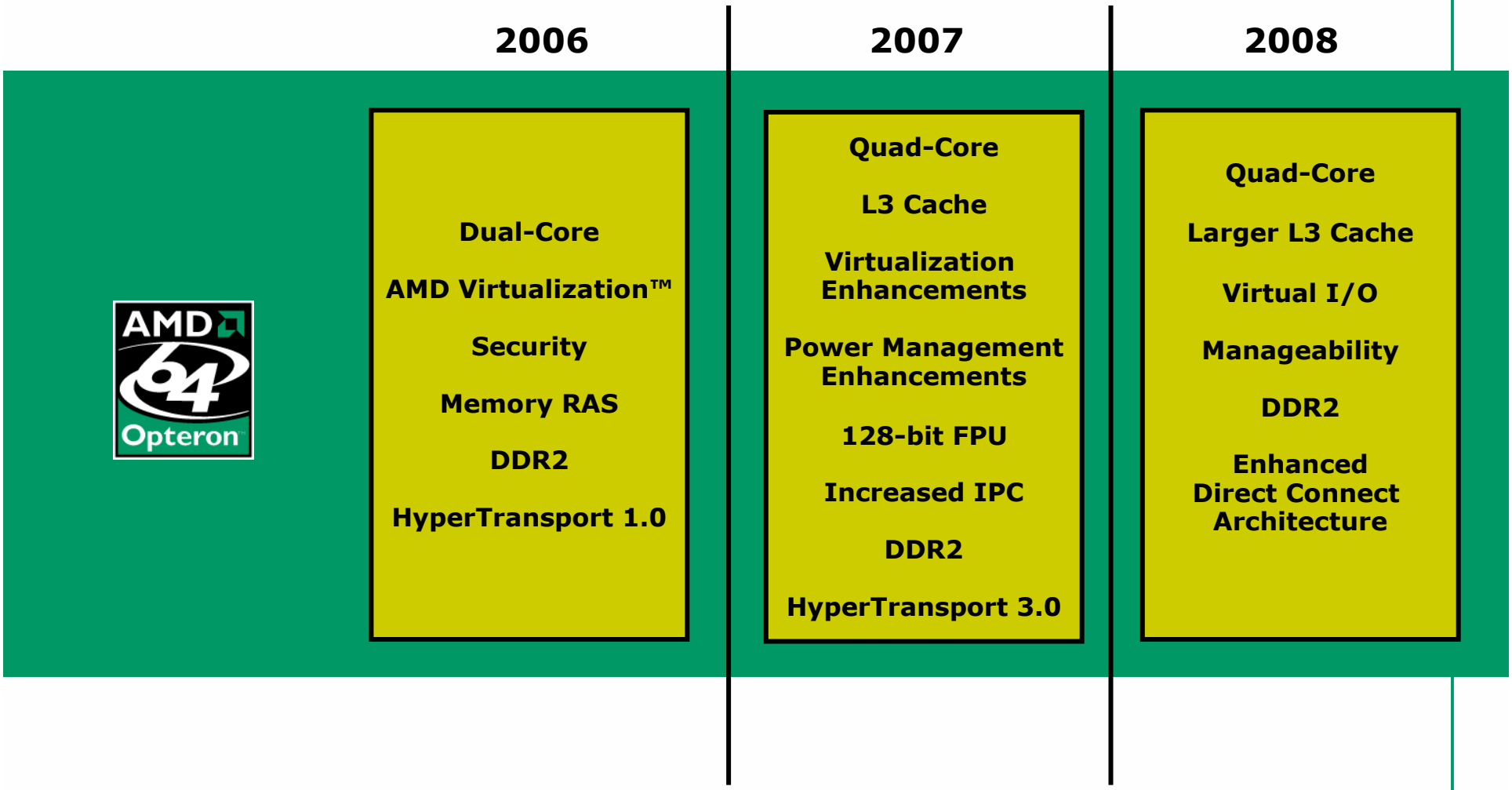
AMD
Smarter Choice



AMD
Smarter Choice

Backup slides

Technologies Roadmap: Server/Workstation



Core "SVM" Architecture: Intercepts

AMD
Smarter Choice

Guest runs until:

- It performs an action that causes an exit to the host
- It explicitly executes the `VMMCALL` instruction

The VMCB for a guest has settings that determine what actions cause the guest to exit to host

- These intercepts can vary from guest to guest
- Two kinds of intercepts
 - Exception & Interrupt Intercepts
 - Instruction Intercepts
- Rich set of intercepts allow the host to set customize each guest's privileges

Information about the intercepted event is put into the VMCB on exit

Core SVM Architecture: VMRUN

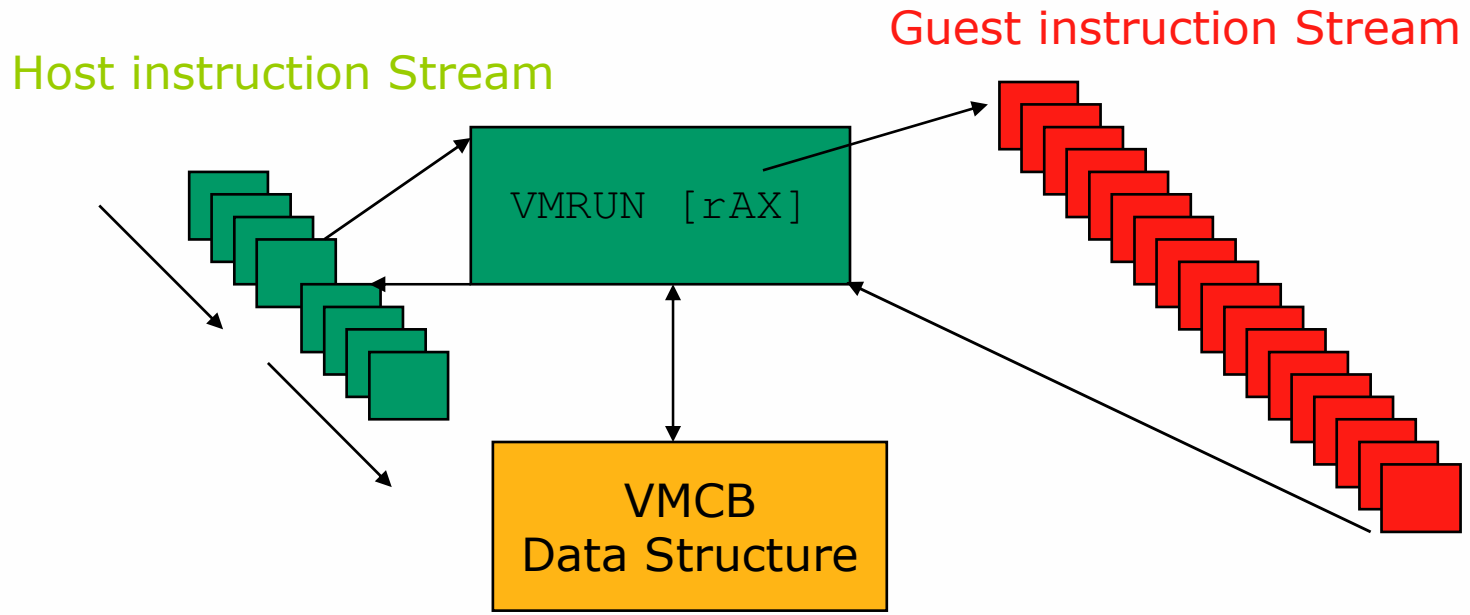
Virtualization based on Virtual Machine Run (VMRUN) instruction

VMRUN executed by host causes the guest to run

Guest runs until it exits back to the host

World-switch: host → guest → host

Host resumes at the instruction following VMRUN



Core "SVM" Architecture: VMRUN

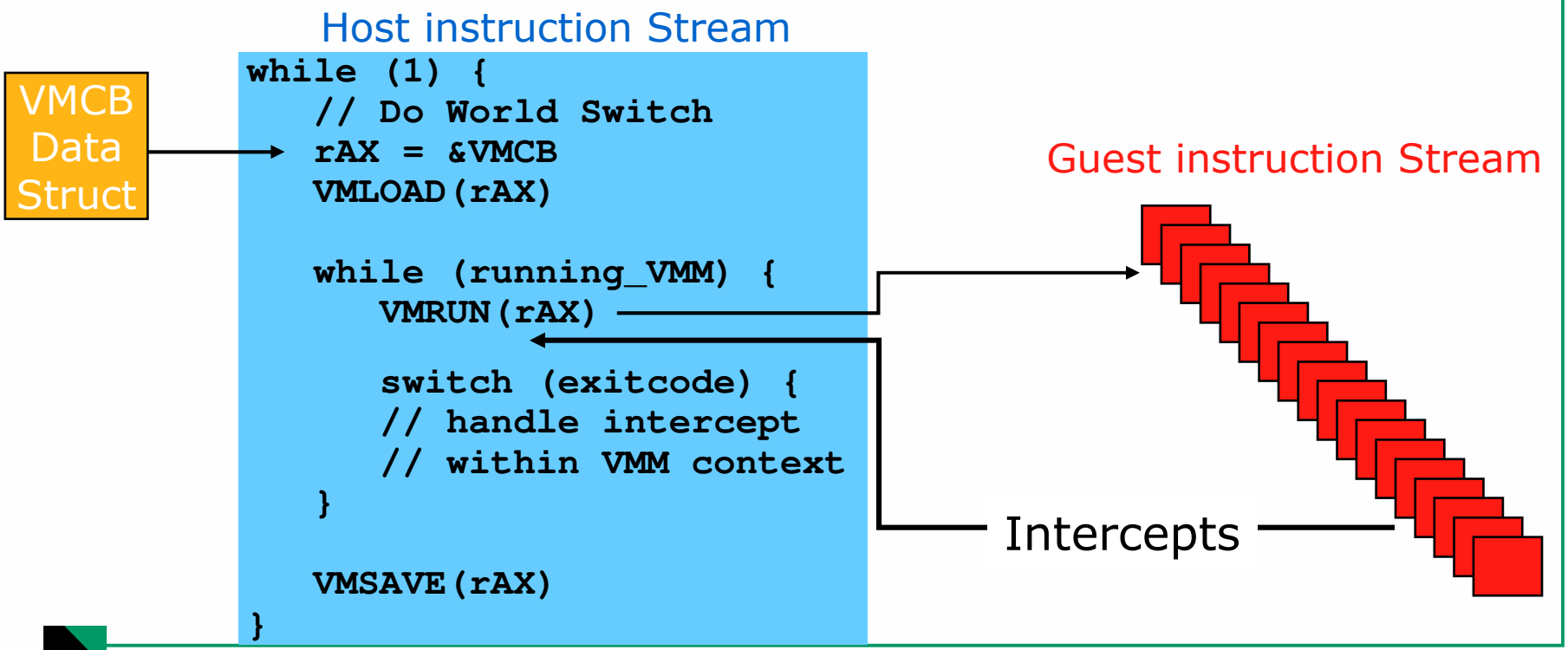
AMD Smarter Choice

Virtualization based on Virtual Machine Run (VMRUN) instr.

VMRUN executed by host causes guest to run in "Guest Mode"

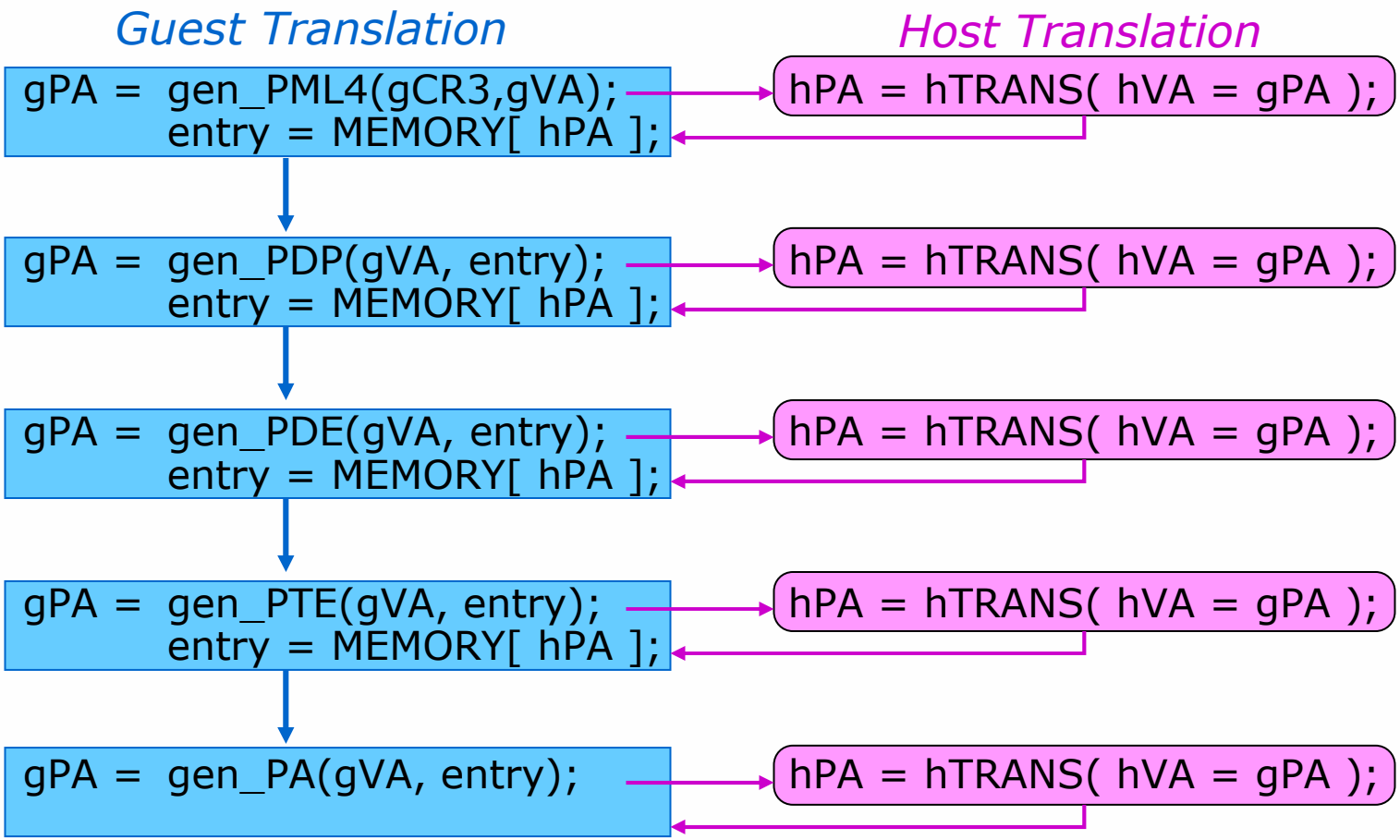
Guest runs until it exits back to the host

Host resumes at the instruction following VMRUN



Nested Paging: CPU walks both Guest & Host Page Tables

- CPU maps each Guest_PA to Host_VA and then translates to Host_PA
- CPU builds compound gVA_to_hPA TLB entries (guarded by ASID)
- Far more efficient than "Shadow Page Tables", all handled by CPU



CodeAnalyst for Windows

AMD
Smarter Choice

Major functionality on Windows:

Timer-based profiling (TBP)

- On APIC enabled systems, the finest time resolution is 0.1ms
- On non-APIC enabled systems, the finest time resolution 1.0ms.

Event-based profiling (EBP)

- Designed to profile the all 32 performance events of Athlon
- Designed to profile the all 104 performance events on Opteron
- With event multiplexing technique, CodeAnalyst can profile more than 4 events simultaneously on AMD hardware.
- Meaningful Event counts: i.e. IPC, cache miss rate etc.

Call Stack Sampling:

- Unwind the call stack to collect call path data on hotspots and display caller/called relationship on hotspots during post process.

Thread Analysis

- Identify thread numbers, affinity, and CPU activity
- Identify remote memory access.

Trace generation and Pipeline Simulation provided

- Graphical display of execution pipeline events with penalty and dependency annotations (for Windows environment only at this time)

CodeAnalyst for Windows (2)

Other general functionalities:

Low overhead, system-wide profiling
- Can profile dynamically loaded modules
- Can profile kernel mode drivers as well as applications

Supports multiprocessor profiling

Support on Microsoft Operating Systems:
- Windows NT, Windows 2000
- 32/64 bit Windows XP, Windows Server 2003
- 32/64 bit Windows Vista, Longhorn

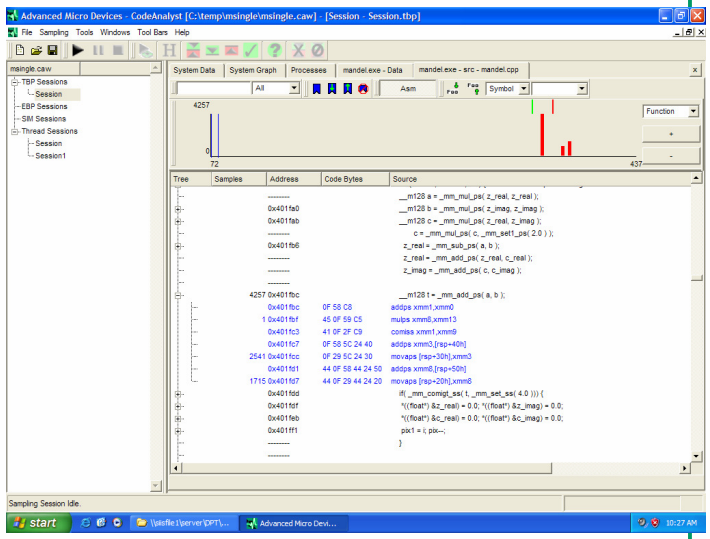
GUI to ease the performance analysis

Visual Studio 2005 integration

Command line utilities
- Supports execution scripts

Supports profiling of JIT code
- Java
- Microsoft .NET
- Driver JIT code

Profile control API library
- Pause and resume profiling



CodeAnalyst for Linux

Built on top of the Oprofile driver and daemon

Modified driver and daemon to support Java Profile

Purpose is to provide ease of profile configuration and quicker data analysis

Consistent user interface with CodeAnalyst Windows for cross platform ease of use

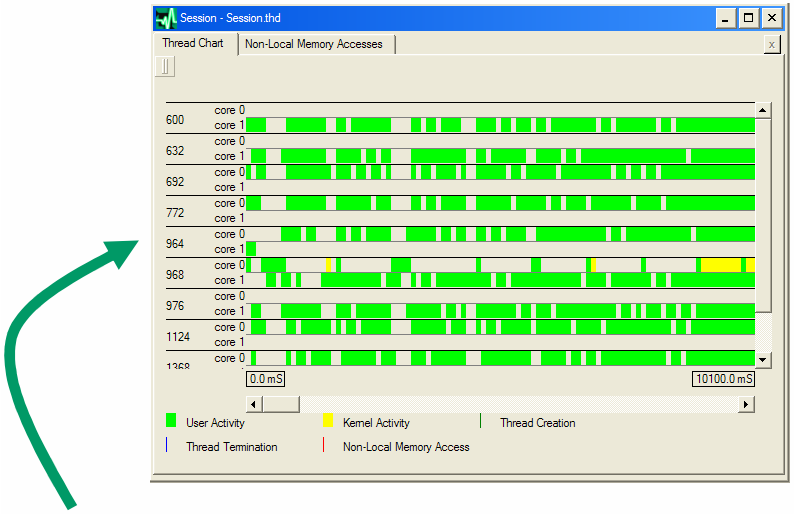
Similar profile data drill down and graphical display as provided in CodeAnalyst Windows

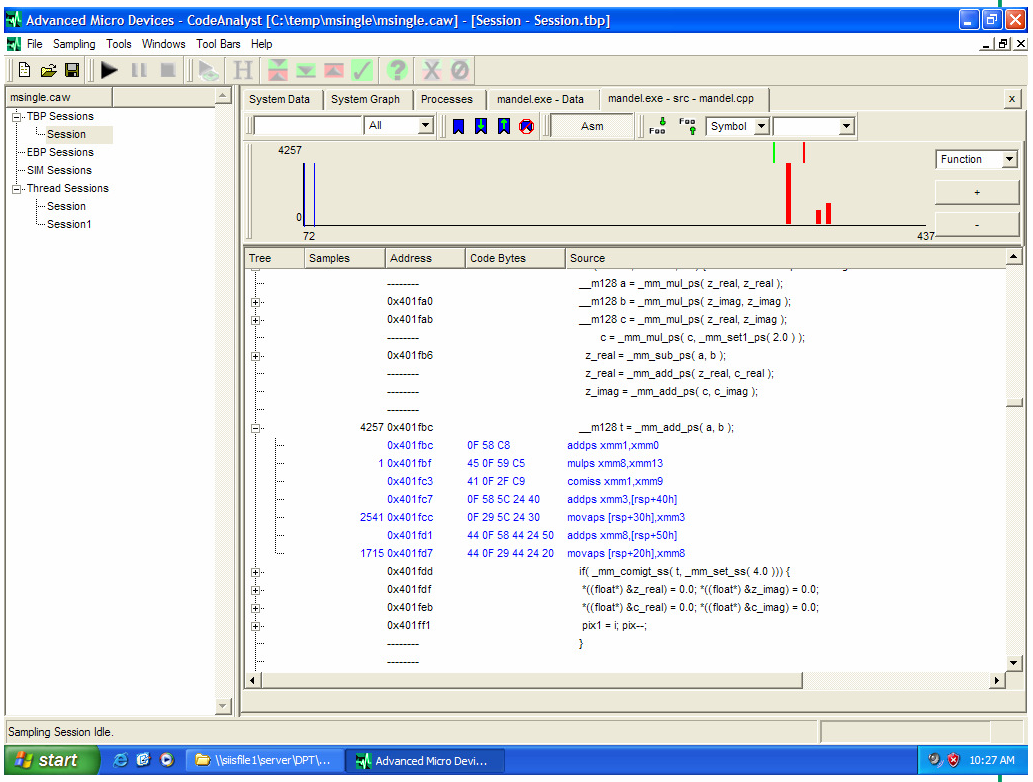
Trace and pipeline simulation are provided as command line utilities in simulation tool kits.

AMD CodeAnalyst™ Profiler

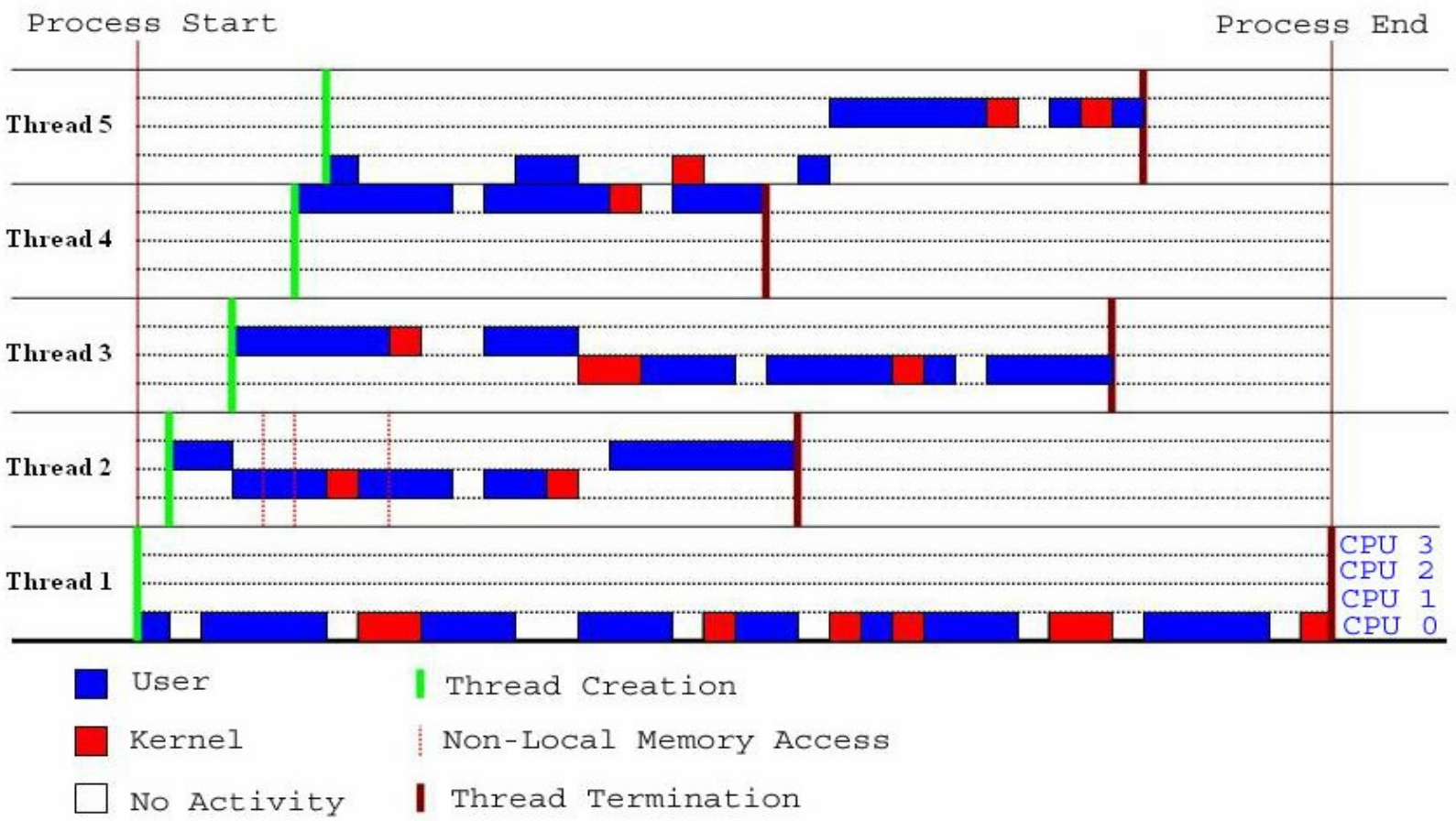
Use AMD CodeAnalyst, profile your 32 and 64-bit code!

- Timer-based & event-based profiler
- Integrates with the VS2005 IDE





CodeAnalyst Thread Analysis Data View



Linux64: GNU Compiler Flags
GCC Flags Complexity (gcc 3.3.3)

AMD
Smarter Choice

- **-O/-O1:**

```
-fdefer-pop -fmerge-constants -fthread-jumps
-floop-optimize -fif-conversion -fif-conversion2
-fdelayed-branch -fguess-branch-probability
-fcprop-registers -fomit-frame-pointer
```

- **-O2:**

```
-O1 -fforce-mem -foptimize-sibling-calls -fstrength-reduce
-fcse-follow-jumps -fcse-skip-blocks -frerun-cse-after-loop
-frerun-loop-opt -fgcse -fgcse-lm -fgcse-sm -fgcse-las
-fdelete-null-pointer-check -fexpensive-Optimisations
-fregmove -fschedule-insns -fschedule-insns2
-fsched-interblock -fsched-spec -fcaller-saves -fpeephole2
-freorder-blocks -freorder-functions -fstrict-aliasing
-funit-at-a-time -falign-functions -falign-jumps
-falign-loops -falign-labels -fcrossjumping
```



- **-O3:**

```
-O2 -finline-functions -fweb -frename-registers
```

Pathscale Compiler Flags - Optimisation Flags

AMD Smarter Choice

Most aggressive: -Ofast

- Equivalent to –O3 –ipa –OPT:Ofast –fno-math-errno

Aggressive : -O3

- Optimisations for highest quality code enabled at cost of compile time
- Some generally beneficial Optimisation included may hurt performance

Reasonable: -O2

- Extensive conservative Optimisations
- Optimisations almost always beneficial
- Faster compile time
- Avoids changes which affect floating point accuracy.

PathScale™

Linux64: PathScale Compiler Flags Functionality Flags

- **-mcmodel=medium**

 use if static data structures are greater than 2GB

- **-ffortran-bounds-check**

 (fortran) check array bounds

- **-shared**

 generate position independent code for calling shared object libraries

- **Feedback Directed Optimisation**

 STEP 0: Compile binary with -fb_create_fbdata

 STEP 1: Run code collect data

 STEP 2: Recompile binary with -fb_opt fbdata

- **-march=(opteron|athlon64|athlon64fx)**

 optimize code for selected platform (Opteron is default)

Linux64: PathScale Compiler Flags
Optimisation Flags

AMD
Smarter Choice

- **Most aggressive: -Ofast**

 Equivalent to -O3 -ipa -OPT:Ofast -fno-math-errno

- **Aggressive: -O3**

 Optimisations for highest quality code enabled at cost of compile time

 Some generally beneficial Optimisation included may hurt performance

- **Reasonable: -O2**

 Extensive conservative Optimisations

 Optimisations almost always beneficial

 Faster compile time

 Avoids changes which affect floating point accuracy.

PathScale™

Linux64: PathScale Compiler for Dual Cores
Controlling CPU Core/Memory Affinity

AMD
Smarter Choice

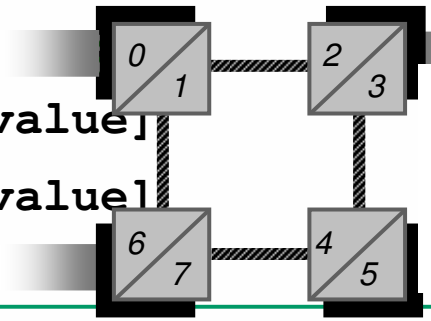
PathScale 2.3 allows users to control process/core affinity in OMP applications when compiled and linked with –mp

- **Environment variables required for OMP and NUMA**

```
> setenv OMP_NUM_THREADS 4
> setenv PSC_OMP_AFFINITY TRUE                    [TRUE/FALSE]
> setenv PSC_OMP_AFFINITY_GLOBAL TRUE             [TRUE/FALSE]
> setenv PSC_OMP_AFFINITY_MAP 4,5,6,7
```

- **Note existence of:**

```
PSC_OMP_CPU_STRIDE [Integer value]
PSC_OMP_CPU_OFFSET [Integer value]
```



PathScale™

Linux64: Absoft Compiler Flags
Optimisation Flags

AMD
Smarter Choice

Below are 3 different sets of recommended Absoft compiler flags for flag mining application source bases:

- **Most aggressive: -O3**

 loop transformations, instruction preference tuning, cache tiling, & SIMD code generation (CG). Generally provides the best performance but may cause compilation failure or slow performance in some cases

 strongly recommended for any single precision source code

- **Middle of the road: -O2**

 enables most options by –O3, including SIMD CG, instruction preferences, common sub-expression elimination, & pipelining and unrolling.

 in double precision source bases a good substitute since Opteron has the same throughput on both scalar and vector code

- **Least aggressive: -O1**

absoft®

Linux64: Absoft Compiler Flags
Functionality Flags

AMD
Smarter Choice

- **-mcmodel=medium**

 use if your application statically allocates a net sum of data structures greater than 2GB

- **-g77**

 enables full compatibility with g77 produced objects and libraries

 (must use this option to link to GNU ACML libraries)

- **-fpic**

 use when linking to shared object (dynamically linked) libraries

- **-safefp**

 performs certain floating point operations in a slower manner that avoids overflow, underflow and assures proper handling of NaNs

absoft®

Linux64: Intel Compiler Flags Optimisation Flags

AMD Smarter Choice

- **-axP -xW -ipo -O3**

 -axP optimizes code for P4 processors with SSE3 support - enables vectorization (-axW)

 -xW optimizes code for P4 & compatible processors

 -ipo enables multifile inter-procedural Optimisation

 -O3 turns on several aggressive Optimisations

- **Feedback Directed Optimisation**

 ***STEP 0**: Compile binary with -prof_gen*

 ***STEP 1**: Run code collect data*

 ***STEP 2**: Recompile binary with -prof_use*

intel Leap ahead™

Linux64

AMD Smarter Choice

# of processors	Compilers	ACML install directory
Single	GNU g77/gcc or compatible	acml3.0.0/gnu64
	GNU gfortran/gcc	acml3.0.0/gfortran64
	PGI pgf77/pgf90/pgcc	acml3.0.0/pgi64
	PathScale pathf90/pathcc	acml3.0.0/pathscale64
Multi	PGI pgf77/pgf90/pgcc	acml3.0.0/pgi64_mp
	PathScale pathf90/pathcc	acml3.0.0/pathscale64_mp

Linux32

AMD Smarter Choice

# of processors	Compiler	SSE supported	ACML install directory
Single	GNU g77/gcc or compat.	SSE & SSE2	acml3.0.0/gnu32
		SSE, no SSE2	acml3.0.0/gnu32_nosse2
		No SSE or SSE2	acml3.0.0/gnu32_nosse
	GNU gfortran/gcc	SSE and SSE2	acml3.0.0/gfortran32
		SSE, no SSE2	acml3.0.0/gfortran32_nosse2
		No SSE or SSE2	acml3.0.0/gfortran32_nosse
	PGI pgf77/pgf90/pgcc	SSE and SSE2	acml3.0.0/pgi32
		SSE, no SSE2	acml3.0.0/pgi32_nosse2
		No SSE or SSE2	acml3.0.0/pgi32_nosse
	PathScale pathf90/pathcc	SSE and SSE2	acml3.0.0/pathscale32
		SSE, no SSE2	acml3.0.0/pathscale32_nosse2
		No SSE or SSE2	acml3.0.0/pathscale32_nosse
Multi	PGI pgf77/pgf90/pgcc	SSE and SSE2	acml3.0.0/pgi32_mp
	PathScale pathf90/pathcc	SSE and SSE2	acml3.0.0/pathscale32_mp

PGI 6.2 Compiler Features

AMD Smarter Choice

Native Auto-parallel F77, F95, HPF, ANSI C & C++ Compilers

Language standards and features

ANSI C99, ISO C++
Fortran 77, Fortran 95, HPF
Inter-language calling
IEEE 754 floating point support
Pragmas to control Optimisations
Optimized ACML 3.0 math library

Platforms and features

X86 32 and AMD64/EM64T code generation
Windows and Linux systems
Unified Binary for AMD64 and EM64T systems
Large File support on 32-bit systems
Byte-swapping I/O

Optimisations

Global Optimisations

- Common sub-expression elimination
- Constant & range propagation
- Copy propagation
- Dead store elimination
- Redundant store elimination
- Tail recursion elimination
- Function inlining
- Profile Feedback Optimisation (PFO)
- Instruction scheduling
- Register allocation
- Flow Optimisations
- Alignment of Loads/Stores/Branch targets

Loop Optimisations

- Induction variable elimination
- Strength reduction
- Nested Loop vectorization
- Nested Loop parallelization
- Loop fusion
- Loop invariance removal
- Loop invariant conditional hoisting
- Loop interchange
- Loop splitting
- Loop Unrolling and jamming
- Loop iteration peeling

PGI 6.2 Compiler Features

AMD Smarter Choice

Native Auto-parallel F77, F95, HPF, ANSI C & C++ Compilers

Multi- thread/core/processor support

Auto-parallelization of loops
- Native threading

OpenMp 2.5

MPICH2

Multimedia support

Vector and scalar SSE/SSE2/SSE3 code generation

Interoperability

Extended gnu Assembler support

ANSI C source and binary compatibility

Productivity

Fast compile times

Compile-time Optimisation listings

PGI Visual Fortran (PVF) 6.2 Edition

Tools Feature

Visual studio 2005 IDE
- Full integration provides familiar interfaces and tools
- Online help integrated

PVF build engine
- Automatic management of build dependencies

Fortran aware editor
- Syntax coloring
- Intrinsic tips
- Keyword completion

Optimized for Microsoft Visual Studio

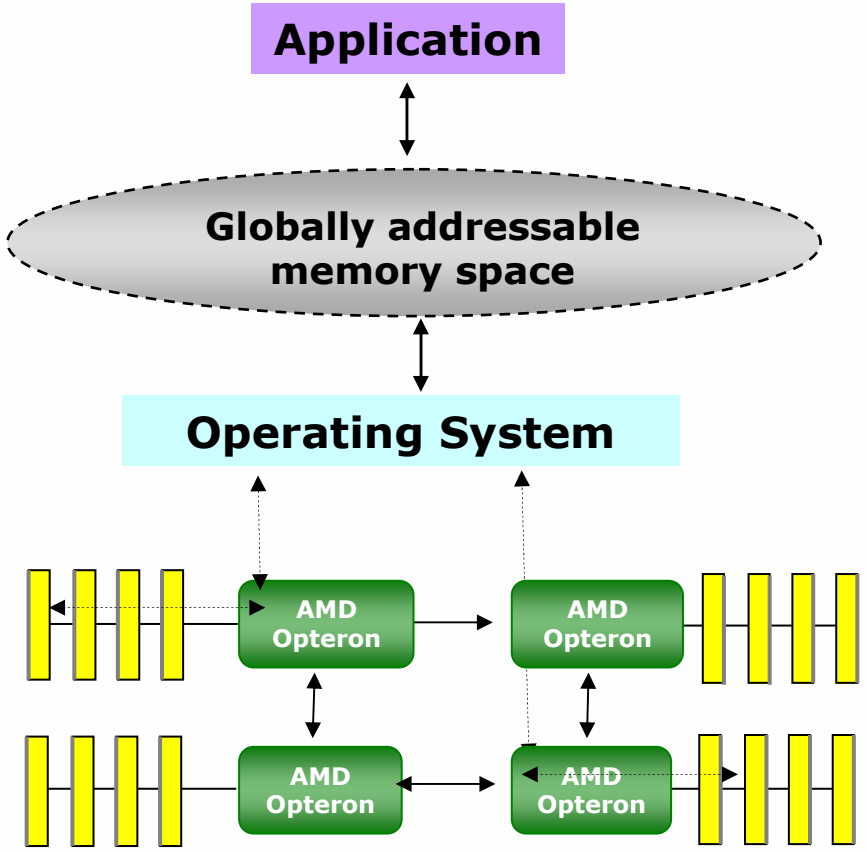
Custom debug engine
- Full featured Fortran debugging
- Visual C++ compatibility
- OpenMP debugging
- PGDBG JIT debug
- PGDBG VC++ debugging
- PGDBG MPICH2 support

CVF and Unix/Linux compatible libraries and modules

Command line compilers, stand alone graphical debugger and profiler

NUMA Architecture *(Non-Uniform Memory Access)*

Provides applications with performance and scalability



Software accesses data belonging to any processor via the global address space

Memory is initialized into globally addressable physical memory space with processors maintaining cache coherency across this space

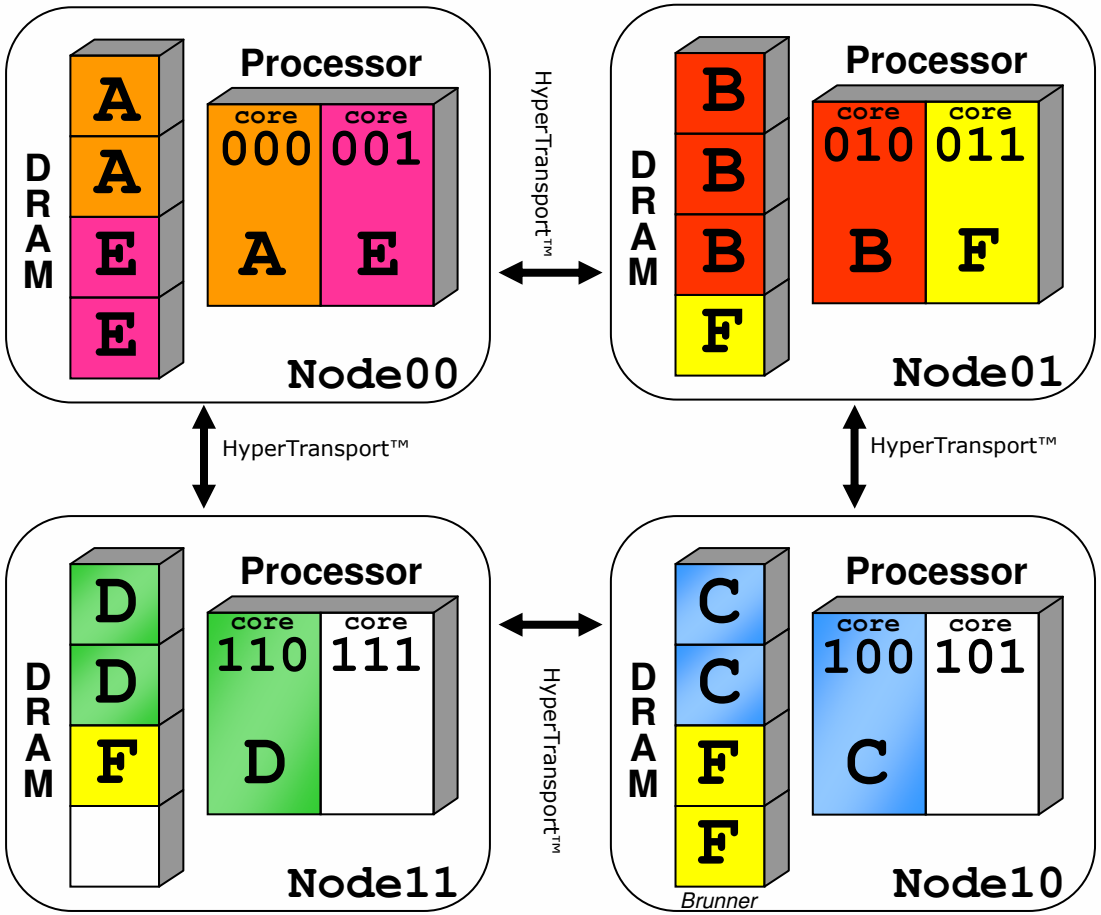
NUMA-aware OS assigns threads from same process to the same NUMA node

Each processor has local memory and uses HyperTransport™ technology for high-speed access to non-local memory

NUMA OS Scheduling Example

Example of an OS optimized for dual core running on an AMD Opteron™ system with 4 dual core processors

- OS distributes threads across processors *first* to minimize contention between threads
- OS attempts to map a thread's memory requests to the local memory of the node (Threads A, B, C, D, E)
- If no memory is left on the current node, OS will use memory of other nodes



AMD64 Porting and Optimizing

AMD
Smarter Choice

Three different ways to get more performance:

Large Memory

- Virtually unlimited address space (256TB)

Multi-threading

- Take advantage of multi-core CPUs

Extra benefits of 64-bit mode

- Improved compiler and libraries
- Twice as many General Purpose Registers (GPRs)